Facultad

de

Ciencias

Sistema inteligente orientado a la visualización y gestión en tiempo real de los turnos, ausencias y descansos de los policías en una Smart City

(Intelligent system oriented to the visualization and management in real time of the shifts, absences and breaks of the police in a Smart City)

Trabajo de Fin de Grado

para acceder al

GRADO EN INGENIERÍA INFORMÁTICA

Autor: Ángel Bolado Carletti

Director: Rafael Menéndez De Llano Rozas

Co-Director: José Miguel Prellezo Gutiérrez

Junio - 2019

TABLA DE CONTENIDO

LISTA DE ACRÓNIOMOS

- AJAX - Asynchronous JavaScript And XML
- API - Application Programming Interface
- BBDD - Base de datos
- BPMN – Bussiness Process Model and Notation
- BSON -Binary Structured Object Notation
- CPU - Central Processing Unit
- CSS - Cascading Style Sheets
- CSV - Comma-Separated Values
- DOM - Document Object Model
- ECMA - European Computer Manufacturers Association
- ETL - Extract, Transform and Load
- FTP - File Transfer Protocol
- GB - Gilgit-Baltistan (Gigabyte)
- HTML - HyperText Markup Language
- HTTP - Hypertext Transfer Protocol
- HTTPS - Hypertext Transfer Protocol Secure
- IDE - Integrated Development Environment
- IEC - International Electrotechnical Commission
- ISO - International Organization for Standardization
- JS - JavaScript
- JSON - JavaScript Object Notation
- LDAP - Lightweight Directory Access Protocol
- MVC - Modelo Vista Controlador
- Oauth - Open Authorization
- PDF - Portable Document Format
- PK - Primary Key
- PM - Policía Municipal
- PNG - Portable Network Graphics
- QA - Aseguramiento de calidad
- RAM - Random Access Memory
- RF - Requisitos Funcionales
- RNF - Requisitos No Funcionales
- SAML - Security Assertion Markup Language
- SPA - Single Page Application
- SQL - Structured Query Language (database query language)
- SSH - Secure Shell
- SSO - Single Sign On
- TB - Terabyte
- TFG - Trabajo final de grado
- URL - Uniform Resource Locator
- XML - eXtensible Markup Language

LISTA DE ILUSTRACIONES

AGRADECIMIENTOS

Con una demora de algunos años, es el momento de zanjar esta fase de mi vida. Tras unos años trabajando en CIC Consulting Informático, me brindaron la oportunidad de finalizar mi vida como estudiante, por eso quería que fuesen los primeros a los que agradecer este fin de ciclo.

Ahora es el momento de agradecer a mis padres la oportunidad que me han brindado de poder ir a la universidad y apoyarme y motivarme en todo momento, sobre todo cuando no veía salida en algunos años de la carrera. Durante esta fase, he tenido la suerte de compartir aula con unos compañeros estupendos, que muchos después se han convertido en amigos. Merece una mención especial mi compañero, y ahora amigo, Víctor que hizo más amena esta andadura.

Fuera de la universidad tuve mucho apoyo del resto de mi familia, motivándome para terminar esta etapa, y mis amigos. También merece una mención especial mi novia, de la que no me ha faltado apoyo y motivación durante estos meses de desarrollo del proyecto.

También quiero dar las gracias a los profesores que he tenido durante mi vida académica, sin los cuales no hubiese llegado hasta donde estoy ahora ni seria la persona que soy ahora mismo.

Por todo esto, **GRACIAS** a todos.

RESUMEN

Durante los últimos años ha aumentado el número de sistemas que recogen información. Este volumen de información desestructurada es necesario tratarla y componer información válida y relevante para los usuarios que van a trabajar con ella.

Estos sistemas cada vez se van introduciendo más en las entidades públicas, que las permiten manejar de forma más eficiente todas sus labores cotidianas, ya sea la de obtener certificados, gestionar plantillas de trabajadores... Todos estos datos están en una base de datos de la cual se puede obtener la información y trabajar con ella, pero para ello se necesita un sistema que sepa interpretar y representar toda esa información.

Así, al cliente le surge la necesidad de informatizar un proceso que actualmente se está llevando a cabo de forma manual. Por tanto, es necesario que, en una plataforma, en la que ya registran incidentes y eventos, y en la que puedan visualizar los medios humanos PM (policías); realizar una serie de acciones sobre los mismos. Lo más importante es dar el paso de llevar la gestión de los descansos de estos policías y pasar del papel y boli a una plataforma informática que les liberen de ciertas tareas, como calcular los excesos de tiempo de descanso. Esta plataforma muestra un gran volumen de datos de una forma organizada, ya que está encuadrada dentro del proyecto que se conoce como Smart City.

A esta plataforma se la puede considerar una herramienta Big Data ya que facilitan las tareas de administración de los recursos de una ciudad. En ella se introducen diariamente los PM que nuestra aplicación lee y con los que trabaja.

Palabras clave: Smart City, Big Data, recursos humanos, Base de datos, Node.js, JavaScript, modelo vista controlador (MVC).

ABSTRACT

In recent years, the number of systems that collect information has increased. This volume of unstructured information is necessary to treat it and compose valid and relevant information for the users who will work with it.

These systems are increasingly being introduced in public entities, which allows them to manage more efficiently all their daily tasks, whether to obtain certificates, manage workers' templates ... All these data are in a database from which you can get the information and work with it, but for this you need a system that can interpret and represent all that information.

Therefore, the client has the need to computerize a process that is currently being carried out manually. They require that, in a platform, in which they already record incidents and events, in which they can visualize the human resources PM (police) and perform a series of actions on them. The most important thing is to take the step of taking the management of breaks, breaks, from these police officers of the paper and pen to a computer platform that will free them from certain tasks, such as calculating the excesses of rest time. This platform shows a large volume of data in an organized way, which is known as Smart City.

In addition to this, it will be necessary to obtain those human resources PM from a system in which they will be inserted daily into the platform so that the data is always accessible from the city's management application, Smart City. This platform can be considered a Big Data tool since it facilitates the administration of the resources of a city.

Keywords: Smart City, Big Data, human resources, Database, Node.js, JavaScript, model view controller (MVC).

1. INTRODUCCIÓN

El primer capítulo explicará el problema que se trata de resolver, así como las motivaciones que han hecho posible su realización y los objetivos que se deben alcanzar a la finalización de este proyecto.

1.1. Motivación y objetivos

Actualmente hay muchos sistemas recogiendo información, ya sea directamente de sensores o de otros sistemas. Todos estos datos, tanto estructurados como no estructurados, son obtenidos y almacenados y requieren de herramientas que faciliten las tareas de administración de estos. Muchas organizaciones están creando o añadiendo a sus infraestructuras herramientas de **Big Data**[2][3][4] para poder llegar a satisfacer los tres grandes retos que suponen esta gran cantidad de información, lo que se conoce como las tres "V" de un sistema Big Data: volumen, velocidad y variedad.

En lo que al proyecto se refiere, el **volumen** se refiere a los datos obtenidos de los diferentes sistemas que, a través de una ETL, que es un software que nos permite obtener datos de API expuestas, transformar esos datos e insertarlos en nuestra base de datos, obtendremos para integrarlos en nuestra plataforma, de una forma estructurada, y así poder trabajar con ellos de una forma más ágil. La **velocidad** es un punto imprescindible en este sistema, ya que se tratan temas críticos, en nuestro caso la policía, pero se da soporte a otros organismos como los bomberos o protección civil. Por eso, es importante que el acceso a la información se realice de la forma más rápida posible, ya que tardar más de lo necesario podría tener consecuencias críticas. Y, por último, la **variedad** de los datos de los que se compone el sistema. Como se mencionó anteriormente, a través de una ETL obtenemos la información de distintos sistemas y sensores, ya sean los horarios de los policías. como explicaremos más adelante, las lecturas de sensores de ruido, la ocupación de estaciones de bicicletas municipales, o hasta la cantidad de residuos que contiene un contenedor de basura.

Por todo esto, surgen las ciudades inteligentes, más conocidas como **Smart Cities**. Sobre esto se va a construir el proyecto, el cual pretende resolver un problema que hasta ahora había en esta entidad pública.

El módulo que se va a desarrollar permitirá administrar a los medios policiales humanos de una ciudad. Hasta ahora, esto se había llevado a cabo de forma manual a través de un Excel. Ese Excel era actualizado diariamente con los medios humanos PM del día, y sobre él se trabajaba. No es que Excel sea una herramienta poco potente y no les permitiese hacer bien su trabajo, pero los

cambios que realizaban sobre el no eran visibles para el resto de los funcionarios que tenían acceso, por lo que se producía una incongruencia de datos. Se pueden modificar los turnos, el tipo de medio policial, los detalles de sus tareas diarias... Otros funcionarios podían llamar a estos que tienen acceso a esta información para realizar alguna consulta, y si hubiese habido cambios, depende a quien llamases, unos responderían una cosa y otros otra. Con este módulo se permite unificar toda la información de forma que los cambios sean visibles para todos los usuarios.

Además, una de las tareas más necesarias de informatizar es la gestión de pausas, descansos, de los medios humanos PM, que hasta ahora se recogían también en el Excel. El proceso de las pausas de definirá en los requisitos funcionales, pero su funcionamiento es principalmente el siguiente: un medio humano PM notifica que va a comenzar la deslocalización, que es cuando se mueven de su puesto al lugar donde hacen el descanso, y se registra en el Excel. Cuando llega al lugar donde harán el descanso vuelven a notificar, esta vez el inicio de la pausa, y se registra también. Cuando terminan la pausa, vuelven a notificar que han acabado y se registra. Parece algo sencillo, pero todo esto llevado en un Excel y teniendo en cuenta que deben estar atentos a que no se pasen del tiempo permitido de deslocalización y de pausa, hacen que las tareas en un Excel se vuelvan un tanto engorrosas.

Por esto se propone hacer este módulo, en el que los medios estarán visibles en una tabla, donde se podrán activar y automáticamente pasarán a otra vista donde se gestionarán estas pausas a través de unos colores que ha definido el organismo. Además, se tendrá que notificar visualmente cuando un medio humano PM se exceda de su tiempo de deslocalización y de pausa.

El desarrollo de este módulo se llevará a cabo en los siguientes capítulos, los cuales se estructuran de la siguiente manera:

- **Capítulo 1**: se hace una introducción al problema y los objetivos que se tienen que cumplir.
- **Capítulo 2**: se especificará la metodología utilizada en el desarrollo del módulo y las herramientas utilizadas para llevarlo a cabo.
- **Capítulo 3**: aquí se presentará el problema y se especificarán los requisitos que debe cumplir el módulo y el sistema que lo albergará.
- **Capítulo 4**: este es el centro del proyecto, donde se define la arquitectura del módulo y del sistema, y se detalla cómo se ha realizado el desarrollo atendiendo a los requisitos especificados en el capítulo anterior.
- **Capítulo 5**: en este se especifican las pruebas que se han realizado para comprobar el correcto funcionamiento del módulo.
- **Capítulo 6**: aquí se hará una conclusión y una exposición de los trabajos futuros

2. METODOLOGÍA Y HERRAMIENTAS

En este capítulo se detallarán las tecnologías y herramientas utilizadas para el desarrollo del módulo que nos compete. Así como la metodología de desarrollo utilizada y la planificación del trabajo.

2.1. Metodología

La metodología utilizada durante el desarrollo de este módulo ha sido la metodología **Iterativa e Incremental**. Con cada iteración se puede tanto incorporar nuevas funcionalidades, como introducir mejoras que complementen las funcionalidades ya implementadas.

En el caso del desarrollo del módulo, el proyecto se divide en **3 iteraciones**: una iteración en la que se obtendrán los datos de un sistema y se introducirán en la base de datos de la plataforma; el desarrollo de la vista en la que se podrán visualizar los medios disponibles para cada día, con las funciones que se detallarán en el siguiente capítulo; y el desarrollo de otra vista en la que se mostraran los medios policiales activos en ese momento para poder hacer la gestión de las pausas.

Cada una de las iteraciones está dividida en varias fases, correspondientes a las etapas de:

- **diseño detallado**: donde se especificarán los requisitos que debe cumplir, las herramientas y metodología que se va a utilizar...
- **implementación**: se detallará el proceso de desarrollo del módulo y de los componentes que lo forman.
- **pruebas** (desarrollo, sistema, aceptación): se llevará a cabo un juego de pruebas que comprueben el correcto funcionamiento de la funcionalidad implementada, atendiendo a los requisitos funcionales definidos.

Durante el desarrollo del proyecto se han mantenido reuniones, presenciales y telefónicas, con la intención de informar sobre los avances y problemas encontrados durante el proceso. En estas reuniones se resolvieron dudas en cuanto a requisitos y funcionalidades con el fin de corroborar el buen entendimiento de estos, ya que algún requisito no estaba muy detallado y podía llevar a confusión y por tanto a una implementación errónea.

Por último, habría que puntualizar, que dando comienzo al proyecto y en alguna iteración se tuvo que realizar un proceso de familiarización y aprendizaje

con herramientas necesarias para el módulo. Como puede ser el caso de la utilización de un software para automatizar flujos de datos entre sistemas, **Apache NiFi** [18].

2.2. Planificación

El desarrollo de este proyecto se lleva a cabo durante un periodo de trabajo dentro de la empresa CIC Consulting Informático, en el que un cliente pide un módulo que debe integrarse en una plataforma ya existente. Este periodo, de unos 6 meses de duración, comenzó el día 19 de noviembre, finalizando el 17 de mayo. Se han dedicado al proyecto aproximadamente 600 horas en total, sin contar el tiempo invertido en la realización de esta memoria.

Como se ha descrito anteriormente, la metodología utilizada ha sido una metodología Iterativa e Incremental. La planificación de las etapas del proyecto incluye las fases descritas en el punto anterior. Además, también se realizó una fase previa en la que se realizó tanto el análisis de requisitos como el diseño del módulo. Como se puede observar en la planificación de las etapas, las iteraciones se dividen a su vez en varias fases.

En la siguiente imagen, *ilustración 1*, se puede observar el diagrama de Gantt con la planificación que se ha realizado para el desarrollo del proyecto. Como se puede ver se planifican tres iteraciones:

- **Iteración 1: Obtener datos medios PM** – Aquí se requiere el desarrollo de un flujo en la herramienta Apache NiFi, con el que se obtengan los medios humanos PM del día siguiente del servicio que tiene el organismo expuesto para que estén disponibles en el módulo antes de que comience el día.
- **Iteración 2: Vista de los medios PM** – Como se puede apreciar se realizan varios desarrollos, desde una tabla en la que mostrar los datos, hasta la posibilidad de editar esos medios en la propia tabla, pasando por el poder aplicar filtros definidos por el cliente. Además, también es necesario que se puedan crear medios humanos PM adicionales a los que se obtienen del servicio del que se obtienen.
- **Iteración 3: Vista gestión pausas PM** – Aquí se hará la gestión de las pausas de los policías que sean activados de la vista anterior. Se creará una tabla por tipo de medio humano PM y se deberá implementar que los cambios surgidos en esta vista sean replicados en todas las sesiones que estén mostrando esta vista en tiempo real.

Task Name	Duration
Módulo gestión pausas PM	**130 days**
Formación inicial	7 days
Análisis de requisitos	5 days
Iteración 1: Obtener datos medios PM	**6 days**
Revision de requisitos	0,5 days
Crear flujo de llamada API	3,5 days
Pruebas	1 day
Documentación	1 day
Iteración 2: Vista de los medios PM	**47 days**
Revisión de requisitos	0,5 days
Revisión de diseño	0,5 days
Desarrollo	**42 days**
Crear tabla con datos actuales	5 days
Añadir filtro de día para visualizar	2 days
Añadir filtros de turno	2 days
Añadir filtro por tipo de medio PM	3 days
Añadir filtro por palabra clave	4 days
Añadir opción para diferenciar medios corrientes de remunerados	2 days
Permitir crear medio PM manualmente	11 days
Añadir opción de editar medio en tabla	6 days
Añadir opción de editar medio en modal	8 days
Pruebas	3 days
Documentación	2 days
Iteración 3: Vista gestión de pausas PM	**65 days**
Revisión de requisitos	0,5 days
Revisión de diseño	0,5 days
Desarrollo	**60 days**
Crear vista con tablas en función del tipo de medio PM	4 days
Rellenar tabla con datos específicos	3 days
Añadir lógica gestión de pausas PM	**45 days**
Posibilidad de marcar cada estado de la pausa	9 days
Crear proceso automático que avise de los excesos de pausa	15 days
Hacer que los cambios sean visibles en tiempo real	22 days
Añadir posibilidad pantalla completa por tabla con las mismas funcionalidades	9 days
Pruebas	3 days
Documentación	2 days

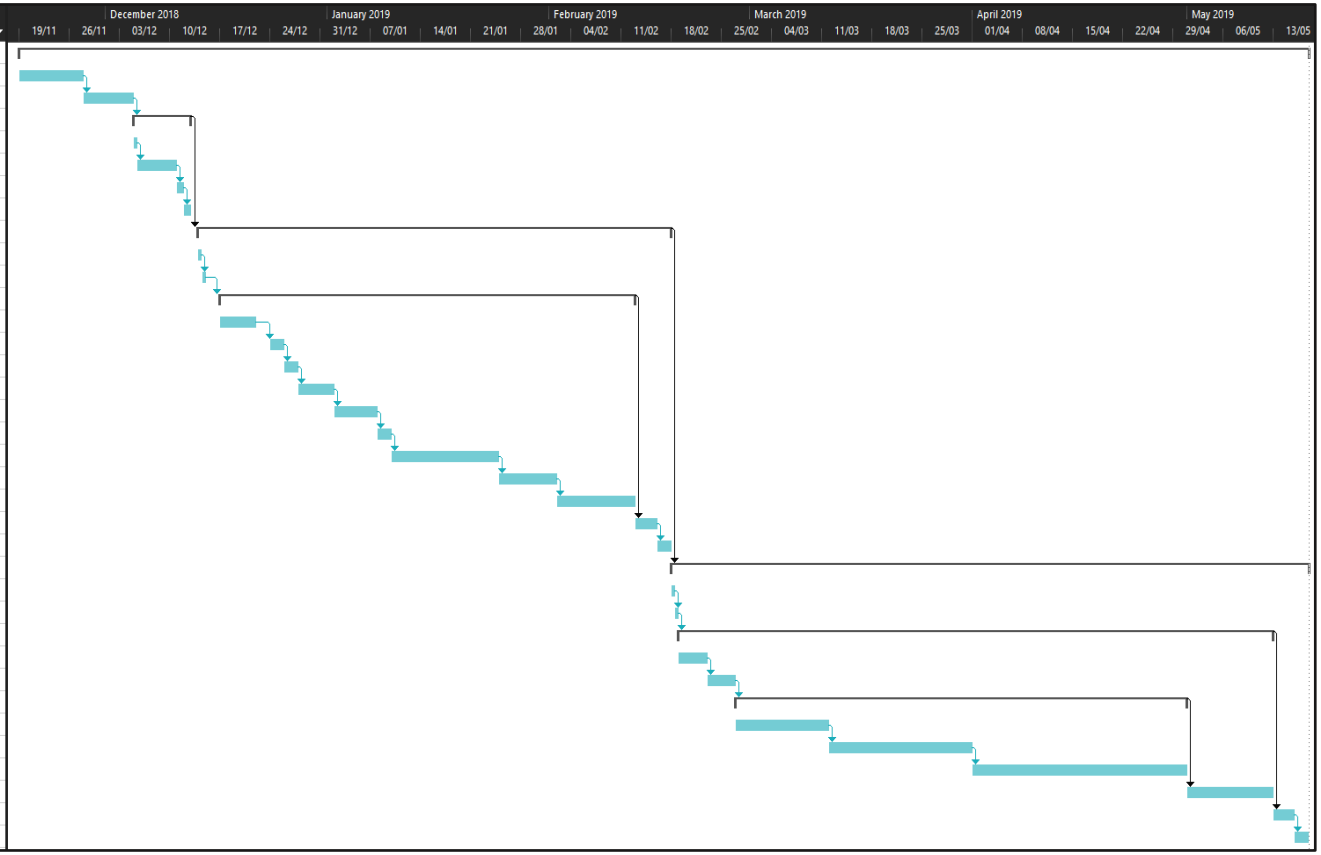

Ilustración 1. Diagrama de Gantt

2.3. Material Utilizado

En este apartado detallaremos las tecnologías y herramientas utilizadas durante el desarrollo del módulo solicitado y la justificación de porque se han elegido estas.

A. Tecnologías

Algunas de las tecnologías seleccionadas, como son HTML5, CSS3 y JavaScript, han sido elegidas al tratarse de una aplicación web. El uso de node.js y express.js para desarrollar la parte de servidor se debe al poder de escalabilidad que tiene esta tecnología, al poder desarrollar módulos independientes e incorporarlos al sistema sin apenas esfuerzos. Se trata de un sistema de gestión para Smart Cities, que puede ser reutilizado para ser implementado en otra ciudad sin apenas cambios.

El uso de ***Orion Context Broker***[5][6][7] se debe a que es un Fiware muy utilizado en el ámbito de las Smart Cities, ya que es *open source* y tiene funcionalidades que facilitan el desarrollo de algunos requisitos. Una funcionalidad muy importante de este Fiware es que cuando se insertan datos de lecturas de sensores, de ruido, por ejemplo, y se requiere que cuando un sensor tenga una lectura fuera de un rango permitido, se desencadene una serie de acciones, esto lo monitoriza el *context broker* de forma que es transparente para el programador y reduce la carga de trabajo. La utilización de MongoDB como base de datos viene impuesta por la utilización y este Fiware.

HTML5[10]

HTML es un lenguaje de marcas que define únicamente el contenido de una página web, pero no su funcionalidad. Estas marcas son interpretadas por los navegadores que muestran la página tal y como se ha especificado.

HTML5 es una evolución de HTML a la que se le añaden nuevas marcas que permiten de forma nativa mostrar contenido que antes solo era posible usar a través de librerías externas. Por ejemplo, reproducir audio, geoposicionamiento...

Ilustración 2. Logotipo de HTML5

CSS3[12]

CSS es el lenguaje que se ocupa de describir la presentación de documentos HTML o XML y de cómo deben ser renderizados estos elementos en la pantalla. Se ha utilizado la versión 3 de este tipo de lenguaje ya que es más potente que las anteriores y contiene funcionalidades nuevas. Por ejemplo, efectos de sombra, animaciones, transiciones...

Ilustración 3. Logotipo de CSS3

JavaScript[11]

JavaScript es un lenguaje ligero orientado a objetos que se utiliza para dar dinamismo a las páginas web y que soportan todos los navegadores actuales y muchos navegadores antiguos. Aparte se utiliza también en otros entornos sin navegador, tales como, Node.js, Apache CouchDB y Adobe Acrobat.

Ilustración 4. Logotipo de JavaScript

Node.js[15]

Node.js es un entorno de ejecución multiplataforma para JavaScript basado en el motor V8 de Google, el mismo que se utiliza en sus navegadores. Fue concebido como un entorno de ejecución de JavaScript orientado a eventos asíncronos que permite construir aplicaciones en red escalables gracias a su diseño. No realiza operaciones de entrada y salida por lo que no es un lenguaje bloqueante. Aunque no se comunique con el Sistema Operativo, está diseñado sin hilos, no significa que no se puedan utilizar

Ilustración 5. Logotipo de Node.js

todos los *cores* del sistema, ya que, a través de código, se pueden crear procesos hijos que pueden comunicarse fácilmente con el proceso principal y así poder utilizar todos los recursos de la CPU que lo va a ejecutar.

Express.js

Express.js es una infraestructura de aplicaciones web para Node.js mínimo y flexible que proporciona un conjunto de características para aplicaciones ya sean web o móviles. Contiene multitud de métodos para HTTP y *middleware* los que permite la creación de una API sólida de forma rápida y sencilla. Alguno de estos métodos que facilitan la tarea de empezar una nueva aplicación son el direccionamiento, conexiones con base de datos, manejo de errores...

Ilustración 6. Logotipo de Express

Orion Context Broker[20]

Orion Context Broker es un importante componente que permite administrar el ciclo de vida de los datos en el conjunto Fiware creado por Telefónica. Así, a través de este *context broker* es posible tanto añadir, actualizar y consultar entidades de contexto, como suscribirse a la información de contexto, entidades con reglas que se deben cumplir para los datos que se inserten, que ejecutaría una acción cuando una modificación lo requiriese.

Ilustración 7. Logotipo de Orion

MongoDB

MongoDB es un sistema de base de datos NoSQL, no relacional, orientado a documentos y de código abierto. MongoDB almacena los datos en documentos flexibles, BSON, similares a los JSON, lo que significa que, de un documento a otro, los campos pueden variar y con el tiempo la estructura de los datos puede cambiar, al contrario que en las bases de datos relacionales.

Ilustración 8. Logotipo de MongoDB

JSON

JSON es un formato de texto completamente independiente del lenguaje, pero utiliza convenciones que son conocidas por la mayoría de los lenguajes por lo que se convierte en un tipo de lenguaje ideal para el intercambio de datos entre diferentes máquinas. Además, para las personas, es fácil de escribir y comprender.

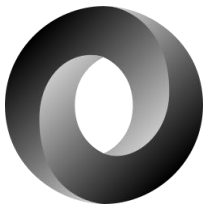

Ilustración 9. Logotipo de JSON

B. Herramientas

Dentro de las herramientas que permiten desplegar aplicaciones node.js, se ha escogido Docker por la facilidad de crear contenedores independientes para las necesidades del sistema. Además, estos enlaces se comunican entre sí a través de sus nombres lo que facilita el desarrollo al no tener que exponer los contenedores que no sea necesario su acceso público. Esta herramienta, Docker, se maneja desde línea de comandos, y para hacer las tareas de mantenimiento y actualización de contenedores se ha escogido la herramienta Rancher, que nos facilita una interfaz gráfica para poder gestionar estos contendores.

Era necesario disponer de una herramienta que permita obtener datos de diferentes sistemas, transforme esos datos a la especificación NGSIv2 con la que trabaja el Fiware y así poder almacenarlos en la base de datos con la que trabaja. Se barajaron más opciones como Kettle, una herramienta ETL de Pentaho, pero la curva de aprendizaje era mayor y Apache NiFi, al estar creado en Java, ofrece la posibilidad de crearte tus propios procesos e implantarlos para poder trabajar con él.

Visual Studio Code

Visual Studio Code es un editor de código fuente ligero y potente desarrollado por Microsoft. Este IDE es perfecto para trabajar con JavaScript, lo que es una buena opción para proyectos como éste, en el que el 90% del código

Ilustración 10. Logotipo de Visual Studio Code

está escrito en este lenguaje. Además, tiene embebido un sistema de control de versiones GIT muy importante cuando realizamos desarrollos importantes.

WinSCP[17]

WinSCP es un cliente de FTP de código abierto, cuya finalidad es la de conectarnos a un servidor remoto y acceder a sus directorios de forma que podamos añadir ficheros locales o editar o eliminar archivos que se encuentran en el servido remoto, al que debemos acceder con un usuario y una contraseña a través de una VPN.

Ilustración 11. Logotipo de WinSCP

Git

Git es un sistema de control de versiones distribuido de código abierto diseñado para manejar tanto proyectos pequeños como muy grandes, con rapidez y eficiencia. Permite mantener un historial de cambios de los diferentes ficheros de la aplicación y lo más importante es que es distribuido, por lo que se puede acceder al código fuente desde cualquier máquina.

Ilustración 12. Logotipo de Git

PuTTY[14]

PuTTY es un cliente SSH y telnet de código abierto con el que se realizan conexiones a servidores remotos.

Ilustración 13. Logotipo de PuTTY

Docker

Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones en contenedores software. Un contenedor es una unidad de software que empaqueta el código y todas sus dependencias para que la aplicación se ejecute de forma rápida y confiable en cualquier entorno informático.

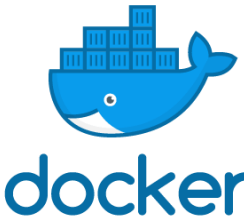

Ilustración 14. Logotipo de Docker

Apache NiFi

Apache NiFi es una ETL, que permite mover datos de múltiples fuentes, reformatearlos, limpiarlos y cargarlos en otra base de datos para ser analizados o apoyar un proceso de negocio. Se basa en un modelo de programación basado en flujo y ofrece la capacidad de operar dentro de clústeres y crear procesos propios por los usuarios para satisfacer necesidades que no ofrece nativamente.

Ilustración 15. Logotipo de Apache NiFi

Rancher

Rancher es una plataforma *open source* que proporciona una interfaz con la que desplegar contenedores Docker con aplicaciones en cualquier infraestructura. Con ella podemos crear nuevos contenedores y administrar los existentes.

Ilustración 16. Logotipo de Rancher

Dia

Dia es un software libre que nos brinda un entorno de modelado con el que poder crear diferentes tipos de diagramas: casos de uso, diagramas de estado...

Ilustración 17. Logotipo de Dia

Office

Office es una suite ofimática que nos proporciona herramientas con las que poder crear todo tipo de documentos, hojas de cálculo, bases de datos...

Ilustración 18. Logotipo de Microsoft Office

3. PRESENTACIÓN DEL PROBLEMA Y ANÁLISIS DE REQUISITOS

Tras haber expuesto los objetivos, en este capítulo se detallan los requisitos software que debe cumplir el proyecto. Una vez establecidos se detallarán tanto los requisitos funcionales como los no funcionales.

3.1. Presentación del problema

Como se ha comentado anteriormente, el módulo de gestión de PM se compone de dos funcionalidades: una permite mostrar los datos recibidos y trabajar sobre ellos; y otra permite que los PM sean visibles diariamente de forma automática. A continuación, se describe cada una de estas funcionalidades.

Obtener datos de los medios humanos PM

Este es el eje principal del módulo, sin el cual ninguno de los siguientes puntos sería posible que se pudiera ejecutar. Este módulo utiliza una API, que la organización tiene expuesta con los medios PM para la fecha que sea necesario consultar, en este caso se consultará siempre la del día siguiente. Esto se hace a partir de una hora especificada por el cliente en la cual los medios humanos se saben que ya están disponibles, así que se planifica el flujo para que estos sean insertados en la plataforma y se pueda trabajar con ellos. Estas acciones están definidas en los requisitos funcionales disponibles en el siguiente apartado.

Vista de los medios humanos PM

Aquí se representan, mediante tablas, la lista de medios humanos PM disponibles para el día seleccionado. Los datos que se van a mostrar son los obtenidos por el apartado anterior. En esta vista también será posible realizar diferentes acciones sobre la tabla y los datos, como, por ejemplo: editar atributos de los medios que se están visualizando, ver medios humanos PM de otro día, filtrar por tipo de medio humano, por turno, por palabra clave...

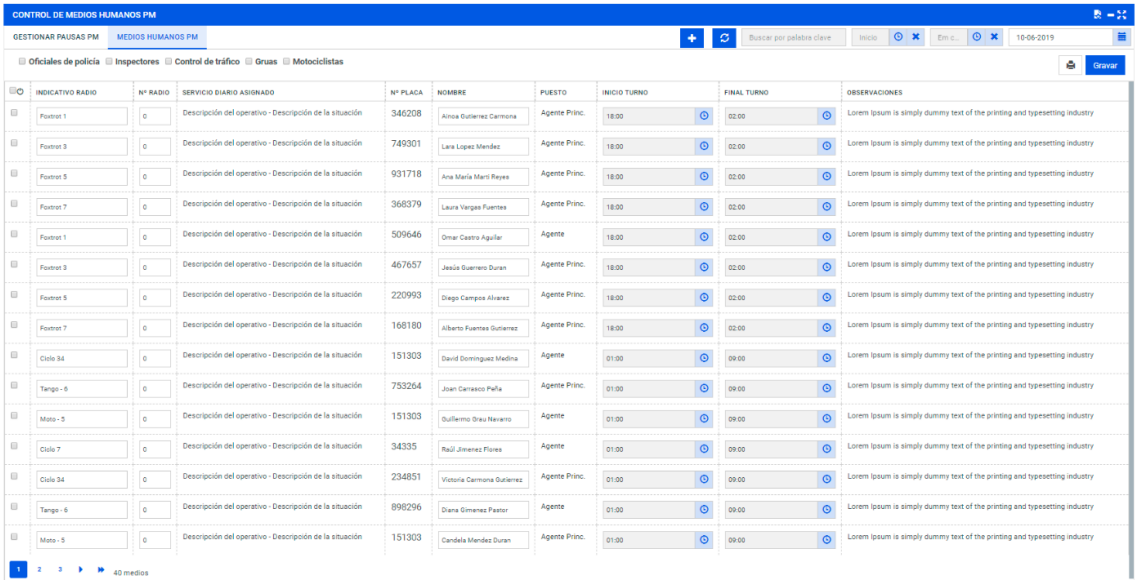

Ilustración 19. Vista medios humanos.

Vista gestión de pausas de medios humanos PM

Esta vista es la finalidad del proyecto, aquí lo que se representan son los medios humanos activos. Estos, se activan desde la vista anterior y debe hacerse manualmente. La principal función de esta vista es la de gestionar las pausas de los medios humanos PM activos. De esta forma se seguirá la secuencia descrita posteriormente. A grandes rasgos, el medio humano PM avisa de que va a iniciar la deslocalización, cuando llega al lugar en el que realiza la pausa avisa y comienza la pausa. Cuando termina, vuelve a notificar que finaliza la pausa y se procede a su finalización.

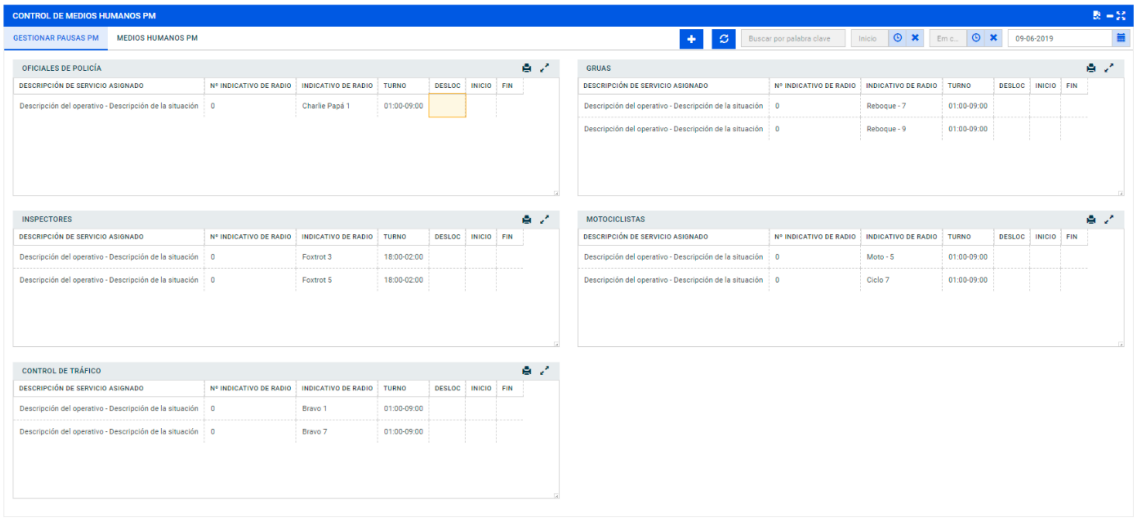

Ilustración 20. Vista gestión pausas PM.

3.2. Requisitos funcionales

Ahora que se ha estudiado el sistema actual, se procede a definir los requisitos que debe cumplir el módulo a desarrollar.

Los requisitos funcionales (RF) detallan lo que un sistema, o módulo de un sistema, debe ser capaz de realizar de una forma óptima.

Los requisitos funcionales definidos para esté módulo, se centran en la forma en la que debe funcionar éste. Además de en especificar todas las funciones que debe cumplir en función al diagrama de actividad Para poder administrar estas pausas, hay que desarrollar diferentes vistas que se especifican en los requisitos. Éstos se dividen en: funciones generales de la plataforma, filtros, tabla de control de medios PM y la tabla de control para el proceso de las pausas de los medios PM.

En las siguientes tablas se detallan los requisitos funcionales identificados para el módulo a desarrollar, identificados por código, nombre de la vista a desarrollar y tipo de funcionalidad.

Código	**RF001**	**Control de medios PM**
		Funciones generales I
Requisito	Posibilidad de visualizar las escalas de los medios PM y activarlas para el servicio actual.	

Código	**RF002**	**Control de medios PM**
		Funciones generales II
Requisito	Posibilidad de utilizar las siguientes características generales: - Imprimir contenidos con acceso al menú de impresoras y configuración de página. - Exportar/transferir contenido a hoja de cálculo (normas ISO/IEC 26300 y ECMA- 376), imagen (ISO/IEC 15948 - PNG) y PDF/A (estándar ISO - 19005). - Mover hacia arriba y hacia abajo, hacia la izquierda y hacia la derecha, los contenidos activos (*scroll*) - Minimizar / maximizar los contenidos y cada una de sus filas o columnas (en el caso de tablas).	

Código	RF003	Control de medios PM
		Filtros
Requisito	Posibilidad de aplicar los filtros (individual o simultáneamente) sobre el contenido del área de información de control de los medios de PM, por cualquier campo de lo visualizado, incluyendo, seleccionar el intervalo de tiempo (pasado y futuro, siempre que sea posible) para el cual se pretende acceder a la información activa, pudiendo elegir la fecha en la que se desean visualizar los datos (escala).	

Código	RF004	Control de medios PM
		Tabla de control de medios PM I
Requisito	Representación y edición de tablas con lista de los medios humanos de PM, con visualización de los "Medios en Servicio" y que tiene como base la hoja de escalas de servicio que es importada y adaptada automáticamente desde la base de datos de PM.	

Código	RF005	Control de medios PM
		Tabla de control de medios PM II
Requisito	Visualización de "Medios en servicio" con tablas desglosadas por tipo de servicio asignado (policía, agente, agente de tráfico, grúas y motociclistas) incluidos los siguientes campos: - **Descripción** del servicio asignado. - **Número de indicativo de radio**: número de contacto por radio del medio o medios de que se trate. - **Indicativo de radio**: nombre del contacto por radio del medio o medios de que se trate. - **Turno**: Identificación del turno al que el(los) medio(s) está(n) de servicio. - **Situación de pausa**: registro, mediante la información del medio o los medios de servicio, de la pausa durante el turno, integrando las siguientes situaciones: desplazamiento para la pausa, el inicio y el final de la pausa. Este registro se realiza y se visualiza en el tablero con la identificación de la situación de la pausa a la que se encuentra el(los) medio(s). Deberán ser identificados en este espacio del tablero cuando se excede el tiempo predefinido para cada una de estas situaciones.	

Código	RF006	Control de medios PM
		Tabla de control de medios PM IV
Requisito	Realización de la gestión de los medios humanos de PM, a la que se accede automáticamente al inicializar el *dashboard* ,a través de una tabla que contiene todos los campos y contenidos actualizados de la base de datos de las escalas de servicio diario de PM en formato editable y con nuevos campos: - **Activación de los elementos en el tablero**: campo que permite activar o desactivar cada uno de los elementos de forma que sean visibles en el tablero de control de pausas PM. - **Número de indicativo de radio**: número de contacto por radio del medio o medios en cuestión. Es importado de la BD de las escalas de servicio diarias de PM. - **Turno**: Identificación del turno en el que el medio está de servicio. Campo importado de la BD de las escalas de servicio diarias de PM, siendo separado en dos campos independientes, uno para la hora de inicio y otro para la hora de fin permitiendo alteración de cada uno individualmente. - **Indicativo radio**: nombre de contacto por radio del medio o medios de que se trate. - **Observaciones**: Campo importado de BD de las escalas de servicio diarias de PM.	

Como se define en el requisito RF005, los medios se categorizan en función de su tipo, y se visualizan en la pestaña de medios en servicio en su tabla correspondiente para gestionar sus pausas. En la siguiente tabla se especifica el requisito funcional que debe cumplir cada tabla, indicando la lógica que debe cumplir.

Código	RF007	Pausas PM
		Tabla de control
Requisito	**1. Comunicar inicio de desplazamiento para pausa** 1.1. Un operativo de la PM en el terreno que esté realizando un turno que incluya pausa, comunica a la sala de despacho el inicio del desplazamiento para el período de pausa, por medio del correspondiente indicativo de radio. **2. Registrar inicio de desplazamiento para pausa en el área de medios en servicio PM** 2.1. El operador de la sala de despacho de PM registra la situación de desplazamiento para la pausa en el área de pausas de medios PM, en la primera de las tres columnas para registrar la situación de las pausas (color amarillo). 2.2. A partir de este registro la plataforma inicia el recuento del tiempo de desplazamiento y pasados 20 minutos se cambia automáticamente el color del registro (de amarillo a rojo). **3. ¿Es necesario asignar un nuevo servicio?** 3.1. En la sala de despacho de PM surge un nuevo servicio y es necesario enviar los medios operativos para su resolución, por lo que se evalúa si es necesario hacer la asignación del citado servicio al operativo que se encuentra en desplazamiento para el período de pausa. **4. Se informa de asignación de nuevo servicio** 4.1. Esta acción se deriva de la respuesta afirmativa al descrito en el punto 3. 4.2. El operador de la sala de despacho comunica la asignación del nuevo servicio y la necesidad de desplazamiento inmediato al lugar al operativo que se encuentra en desplazamiento para el período de pausa. **5. Se desactiva el registro de desplazamiento para pausa en el área de medios en servicio PM (blanco)** 5.1. En el área de pausas de control de medios PM, el operador desactiva el registro de desplazamiento para pausa efectuado en el paso 2. (de amarillo o rojo pasa a blanco). **6. Se desplaza para la resolución del nuevo servicio asignado** 6.1. El operativo que se encontraba en desplazamiento para el período de pausa recibe el comunicado de que deberá acudir a un nuevo servicio y proceder en consecuencia. **7. Comunica el inicio de la pausa (por indicativo radio)** 7.1. Esta acción se deriva de la respuesta negativa a la pregunta del punto 3. 7.2. Después del desplazamiento para el período de pausa, dentro o fuera de los 20 minutos disponibles para el efecto, el operativo de la PM comunica a la sala de despacho el inicio de la pausa, nuevamente por medio del correspondiente indicativo de radio.	

	8. Registrar el inicio de la pausa en el área de medios en servicio PM 8.1. El operador de la sala de despacho de PM registra la situación de inicio de pausa en el área de medios en servicio PM, en la segunda de las tres columnas para registrar la situación de las pausas (color verde). 8.2. El registro existente en la primera columna de desplazamiento se desactiva automáticamente tras el registro del inicio de la pausa en la segunda columna. (de amarillo o rojo pasa a blanco) 8.3. A partir de este registro, la plataforma deberá poder calcular el tiempo de desplazamiento para la pausa del operativo en cuestión (por diferencia entre los registros de las fechas de inicio de la pausa y del inicio del desplazamiento a la pausa). 8.4. La plataforma también debe iniciar el recuento del tiempo de pausa y pasados 30 minutos se cambia automáticamente el color del registro efectuado en la segunda columna (de verde a rojo). 8.5. Los puntos 3, 4, 5 y 6 pueden repetirse en esta fase de pausa, es decir, incluso en situación de pausa y en casos excepcionales, los operativos pueden ser afectos a un servicio. **9. Comunica fin de pausa (por radio indicativo)** 9.1. El operativo de la PM comunica a la sala de despacho el fin de la pausa, una vez más por medio del correspondiente indicativo de radio. **10. Registra fin de la pausa en el área de medios en servicio PM** 10.1. El operador de la sala de despacho de PM registra la situación de fin de pausa en el área de medios en servicio PM, en la tercera de las tres columnas para registrar la situación de las pausas (color azul). 10.2. El registro existente en la segunda columna de inicio de la pausa se desactiva automáticamente tras el registro del final de la pausa en la segunda columna. (de verde o rojo pasa a blanco) 10.3. A partir de este registro la plataforma deberá poder calcular el tiempo de pausa del operativo en cuestión (por diferencia entre los registros de las fechas del fin de pausa y el inicio de pausa).

En la siguiente imagen, *ilustración 21*, se puede observar un diagrama BPMN que muestra el flujo de la gestión de las pausas de los medios humanos PM.

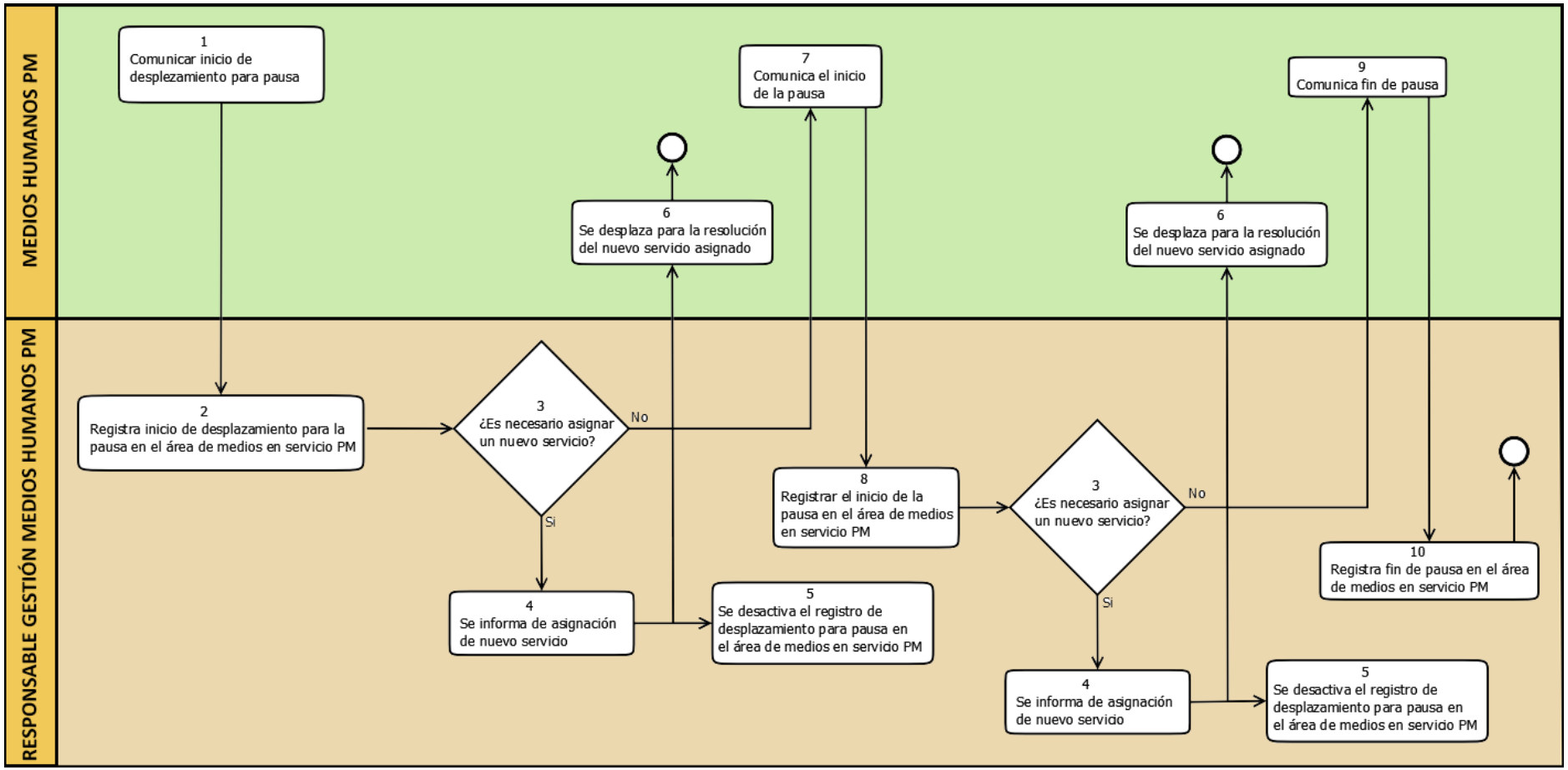

Ilustración 21. Diagrama BPMN de la gestión de pausas

En la siguiente figura se pueden observar los procesos que puede realizar un usuario con acceso a la plataforma y que tenga asignado el rol PM, que es el que le da acceso al módulo que compete este proyecto, ya sea para visualizar medios humanos de diferentes turnos, modificar campos de un medio en concreto y gestionar las pausas de los medios humanos que estén activos en ese momento.

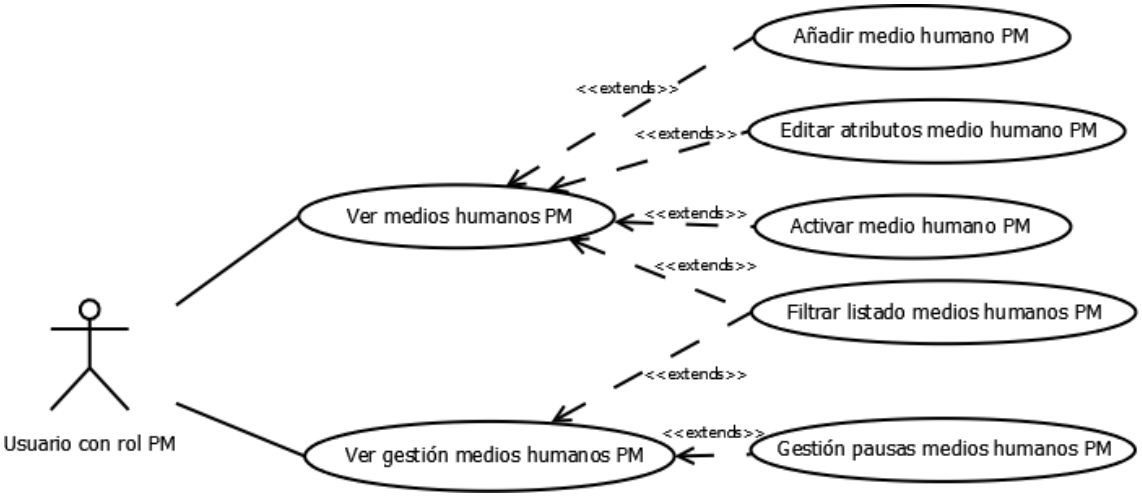

Ilustración 22. Diagrama de casos de uso

Además de este rol, la plataforma maneja otros muchos tipos de usuarios de otros organismos como pueden ser bomberos, usuarios de protección civil, coordinadores, administradores... Cada uno de ellos tiene unas acciones y puede visualizar un contenido diferente.

A parte de lo que compete este proyecto, hay algo que pueden hacer la mayoría de ellos, que es dar de alta eventos e incidentes que ocurren en la ciudad. Para los usuarios de PM, por ejemplo, se pueden asignar a esos eventos e incidentes, medios humanos PM que cargamos a través de la integración necesaria para la implementación de este módulo.

3.3. Requisitos no funcionales

Una vez definidos los requisitos funcionales que se deben, se procede a detallar los requisitos no funcionales que se ha de cumplir la plataforma en la que ha de implantarse el módulo.

Los requisitos no funcionales (RNF), como su nombre indica, no se relacionan directamente con los servicios que el sistema entrega a los usuarios, sino que, describen propiedades o restricciones que debe poseer un sistema.

Los requisitos no funcionales surgen a través de necesidades del usuario, debido a restricciones presupuestas, políticas de la organización, interoperabilidad con otro software o sistemas de hardware... Según esto, los requisitos no funcionales, se pueden dividir en tres grandes grupos: requerimientos del producto, requerimientos de la organización y requerimientos externos. Para definir la plataforma en la que se desarrollará el módulo, esta memoria se centrará en los requerimientos del producto.

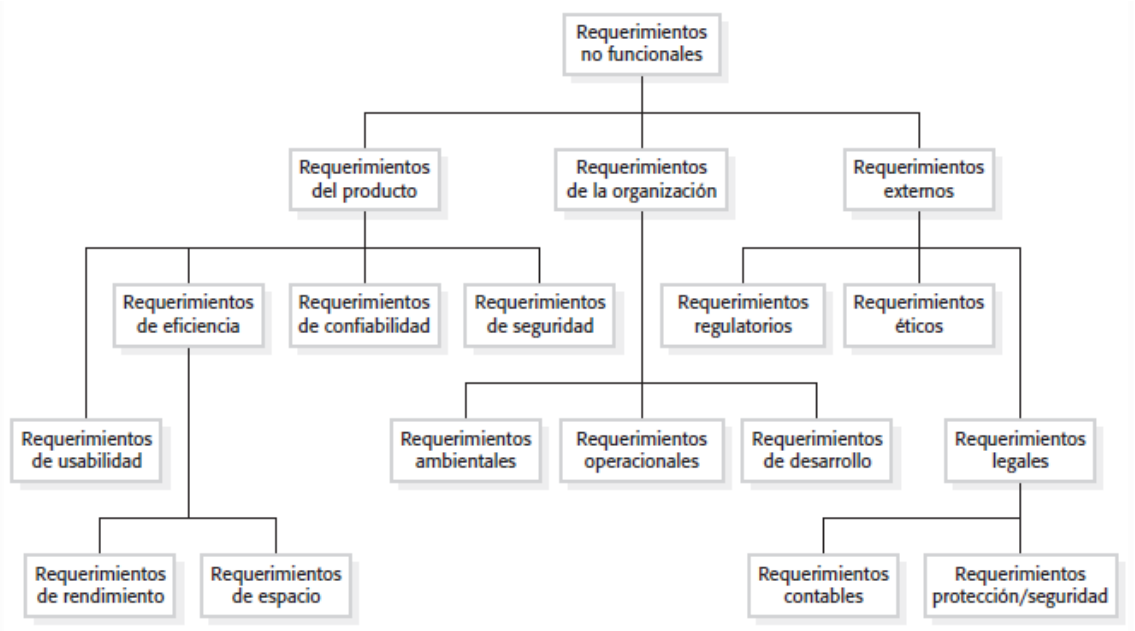

Ilustración 23. Jerarquía de requisitos no funcionales

Las siguientes tablas detallarán los requisitos no funcionales que se encuentran dentro del grupo de los **requerimientos de producto**, los cuales debe cumplir la plataforma en la que se implantará el módulo a desarrollar.

Código	RNF001	Requerimientos de espacio
Requisito	La plataforma suministrada se ejecutará en la infraestructura computacional del organismo con los siguientes recursos: 1. 2x Hosts con 64GB de RAM cada uno, con CPU de 28 *cores*. 2. 50TB de almacenamiento.	

Código	RNF002	Requerimientos de confiabilidad
Requisito	Se tendrá que aplicar una estrategia de tolerancia a fallos, basada en las siguientes premisas: I. Tendrá que ser posible realizar copias de seguridad integrales y parciales sin interrupción de operación. II. Deberá ser posible restablecer las respectivas copias de seguridad, así como hacer disponible el sistema en producción a partir de las mismas copias en un tiempo máximo de 1 hora. III. Todos los procedimientos tendrán que ser efectuados por el equipo de la organización de acuerdo con la documentación proporcionada. IV. Los procedimientos de gestión de fallos deben formalizarse, en particular, implementando los procedimientos de simulación que confirmen la eficiencia de la recuperación de los servicios.	

Código	RNF003	Requerimientos de seguridad I
Requisito	Los accesos de los usuarios tendrán que ser validados utilizando la autenticación a través de uno del siguiente protocolo **SAML.**	

Código	RNF004	Requerimientos de seguridad II
Requisito	1. Los accesos a la plataforma tendrán que ser validados en un directorio LDAP/Active Directory para garantizar mecanismos unificados de autenticación - SSO (Single Sign On). 2. Podrán existir usuarios de la plataforma cuya información de acceso resida localmente en la plataforma. 3. Todos los usuarios serán definidos y gestionados por el	

Código	RNF005	Requerimientos de seguridad III
Requisito	**Acceso a través de HTTP / HTTPS** Las interfaces de usuario disponibles se producirán sobre tecnologías Web, soportando HTTP y su versión segura HTTPS.	

Código	RNF006	Requerimientos de rendimiento
Requisito	La disponibilidad se calcula al final de cada uno de los semestres (entre el 2º y el 6º semestre) mediante la aplicación de la siguiente fórmula: **D [%] = [(P-H + HIP) / P] x100** **P** - Número de horas de funcionamiento potencial en el período considerado; **H** - Número de horas de indisponibilidad en el período considerado; **HIP** - Número de horas de indisponibilidad programada durante el período considerado; La indisponibilidad programada se deriva de la necesidad de ejecución de servicios de mantenimiento, reconfiguración o actualización de los componentes, siempre que se hayan comprobado las condiciones siguientes: 1. La interrupción del funcionamiento no exceda de 30 minutos semanales, o de forma acumulada 2 horas anuales. 2. La interrupción programada de funcionamiento, de conformidad con el apartado anterior, depende de una previa comunicación escrita, dirigida por el proveedor al organismo con una con una antelación mínima de 5 días en relación con la fecha prevista para la interrupción. 3. En caso de indisponibilidad no imputable al adjudicatario, éste se obliga a comunicar por escrito, y demostrar documentalmente, el motivo de estos eventos ante el organismo. 4. Podrán existir otras situaciones excepcionales de intervenciones programadas que, ante el acuerdo previo con el organismo, excedan los tiempos especificados anteriormente. 5. La disponibilidad, calculada para períodos de 6 meses, tendrá que ser superior al 99,5%.	

Código	RNF007	Requerimientos de usabilidad
Requisito	La interacción con los usuarios sigue una implementación multidispositivo (navegador web en sobremesa, portátil y dispositivos móviles: *smartphone*, *tablet*, etc.).	

Código	RNF008	Requerimientos de usabilidad
Requisito	La plataforma tiene que estar preparada para intercambiar información con otros sistemas utilizando: 1. JSON 2. XML 3. CSV	

4. DISEÑO E IMPLEMENTACIÓN

Una vez presentado el problema y definidos los requisitos funcionales y no funcionales que deben cumplir el sistema y el módulo, se detallará el diseño utilizado y la implementación de cada una de las capas de las que el módulo está compuesto. Los siguientes apartados definen el diseño detallado y la implementación de las capas de controladores, negocio y persistencia, además de completarlo con diagramas de despliegue y componentes.

4.1. Diseño Arquitectónico

En este capítulo se va a hacer una descripción de la arquitectura y los componentes que se van a utilizar para llevar a cabo el desarrollo del proyecto.

La arquitectura especifica la estructura y la organización que se va a utilizar en el mismo y compone el diseño a más alto nivel de la estructura de un sistema.

A. Arquitectura del sistema

La plataforma se ha diseñado con una arquitectura basada en capas (ver ilustración 24). Este patrón de desarrollo software está extendido en aplicaciones web y es muy utilizado en el desarrollo de aplicaciones escalables. En este caso, la plataforma está formada por 3 capas que forman el patrón MVC[19]:

- Vista: su principal misión es la de dotar de una interfaz al usuario con un aspecto legible y amigable con la que éste pueda interactuar y realizar las diferentes funciones especificadas en los requisitos. En nuestro caso utilizamos el *framework* AngularJS[16].
- Controlador: esta capa sirve de enlace entre la vista y los modelos. En ella se encuentra toda la lógica de negocio y las peticiones que se realicen desde Vista.
- Modelo: en esta capa se trabaja con los datos, es la encargada de obtener los datos de la base de datos para pasárselos al controlador, o recibirlos de éste para que sean persistentes.

Estas capas están interconectadas entre sí, de forma que puedan pasarse la información que necesiten para resolver una petición del usuario. Esto compone un sistema modular, en el cual puedes generar diferentes versiones independientes de los componentes sin que una modificación en uno afecte a otro.

B. Flujo de control del patrón MVC

A continuación, se detalla la secuencia que se lleva a cabo cuando se realiza una petición en un sistema con el patrón MVC.

1. Un usuario realiza una petición de algún recurso al sistema desde la vista.
2. El controlador recibe la petición y la trata.
3. El controlador notifica al modelo de que se requieren unos datos para mostrar en la vista, o de que es necesario un cambio de estado.
4. El controlador actualiza la vista con los datos que el modelo ha generado.
5. La vista se mantiene a la espera de una nueva iteración por parte del usuario.

C. Diagrama de despliegue

En la siguiente figura, *ilustración 24*, se puede observar el diagrama de despliegue de la plataforma. Se puede observar un servidor central en el cual está instalado NODE.JS que gracias al *framework* ExpressJS tenemos una arquitectura MVC. En este servidor está alojada nuestra aplicación al completo.

También se puede observar que el servidor central se comunica con otro que se llama Fiware, que aloja un Orion, se explicará más adelante, que hace de *middleware* entre el sistema y la base de datos. Esta está alojada en otro servidor que ejecuta la base de datos de MongoDB. La principal función de Orion es la de abstraer al programador de ciertas funcionalidades de las que se encarga este Fiware, como por ejemplo el control de datos para lanzar eventos.

Por último, se puede apreciar un balanceador de carga que redireccionará las peticiones entrantes a nuestro servidor donde se alojan todos los componentes, además que nos permite cumplir con los requisitos no funcionales.

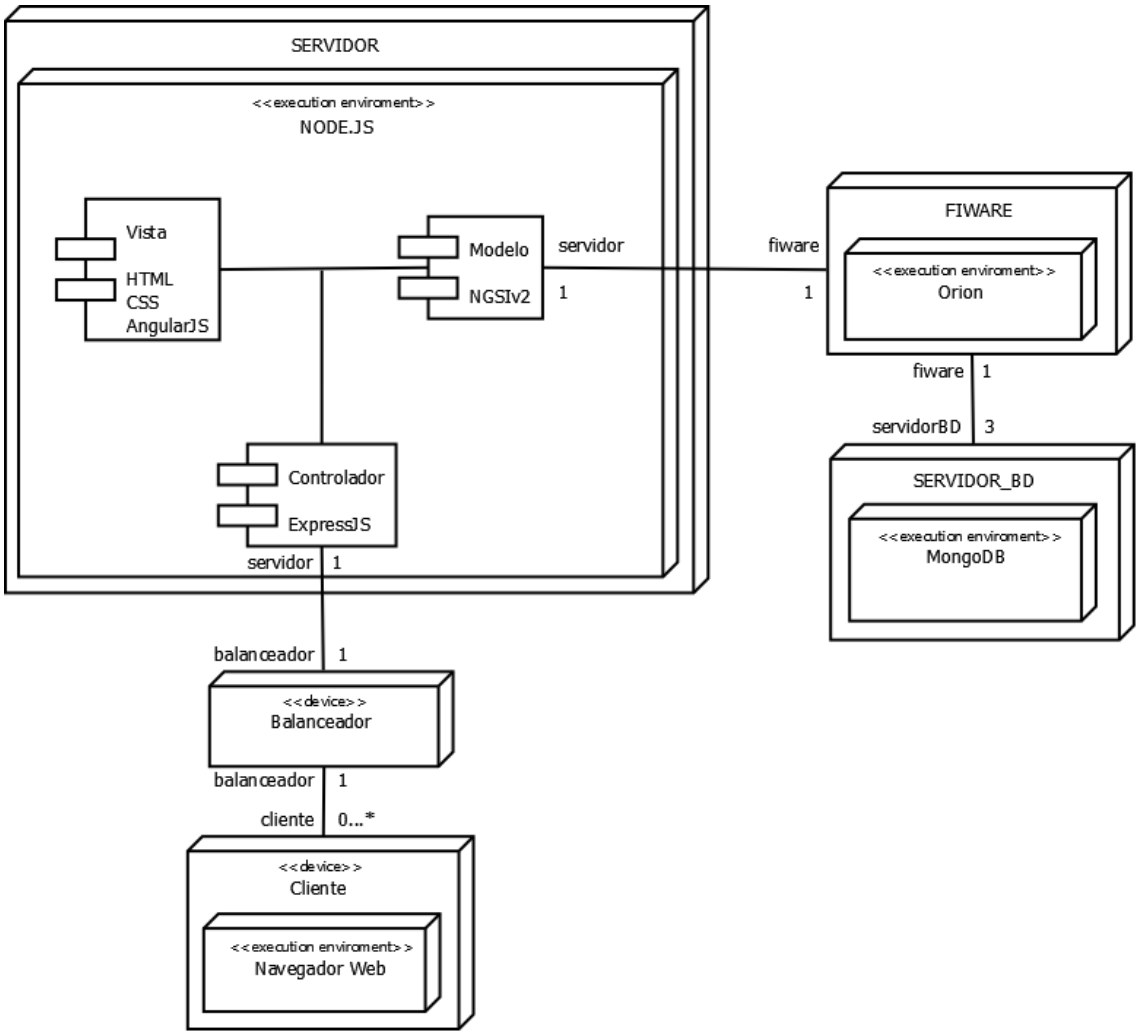

Ilustración 24. Diagrama de despliegue

D. Diagrama de componentes

Este diagrama está simplificado para representar el módulo que compete el proyecto, ya que un diagrama completo de la plataforma sería gigantesco e innecesario en este caso.

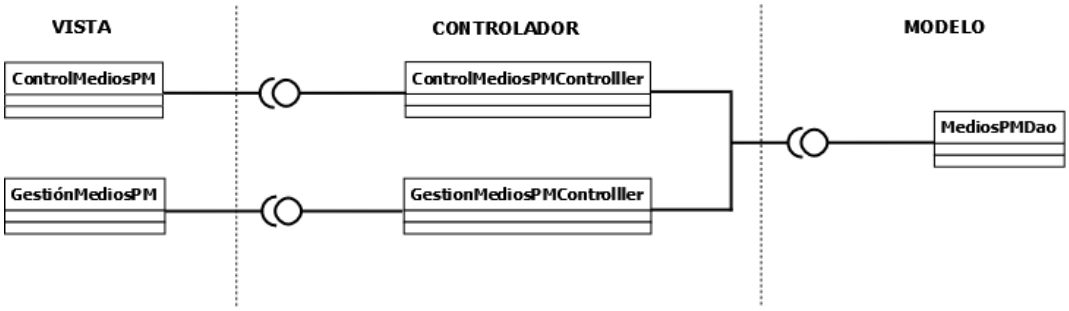

Ilustración 25. Diagrama de componentes

Se pueden observar dos vistas: una referente al listado de los medios humanos PM en la que se podrán editar, activar, y crear nuevos medios humanos PM; la otra vista es la relativa a la gestión de pausas de los medios humanos PM, en la que se gestionarán los descansos que estos realicen durante su turno de trabajo.

Estas vistas, serán gestionadas por dos controladores que albergan la lógica de negocio que hay que implementar para pada una. Por ejemplo, para la primera vista habrá que crear funciones que permitan el filtrado por los campos definidos, guardar los cambios efectuados sobre la tabla, poder editar medios en una ventana modal... El segundo controlador compartirá funciones con el anterior y además las ampliará con las explicitas de la vista. Como son, la activación de la deslocalización, cambiar el estado a inicio de pausa y de esta a fin, planificar tareas para comprobar que el medio humano no se excede de sus tiempos de pausa...

4.2. Implementación

En este apartado hablaremos de la implementación de las diferentes partes del proyecto. Se empezará por la obtención de datos periódicamente desde la ETL Apache NiFi y continuaremos describiendo lo que en cada capa hemos implementado para conseguir cubrir todas las iteraciones definidas en el diagrama de Gantt.

Para empezar con una visión global hay que saber que cada componente está desplegado en un contenedor de Docker. Cada uno con su imagen correspondiente y conectados entre sí para que puedan comunicarse, ya que por defecto los contenedores tienen cerrados todos los puertos, aunque podemos comunicarnos entre contenedores por su alias. Es decir, si el contenedor que ejecuta el Apache NiFi necesita comunicarse con Orion y con la plataforma, se podrán crear enlaces que permitan hacer llamadas a través del nombre del contenedor. En el siguiente punto se explica gráficamente.

A. Apache NiFi

Como se ha comentado en el apartado de las herramientas, Apache NiFi es una ETL, que permite mover datos de múltiples fuentes, reformatearlos, limpiarlos y cargarlos en otra base de datos para ser analizados o apoyar un proceso de negocio. Se basa en un modelo de programación basado en flujo y ofrece la capacidad de operar dentro de clústeres y crear procesos propios por los usuarios para satisfacer necesidades que no ofrece nativamente.

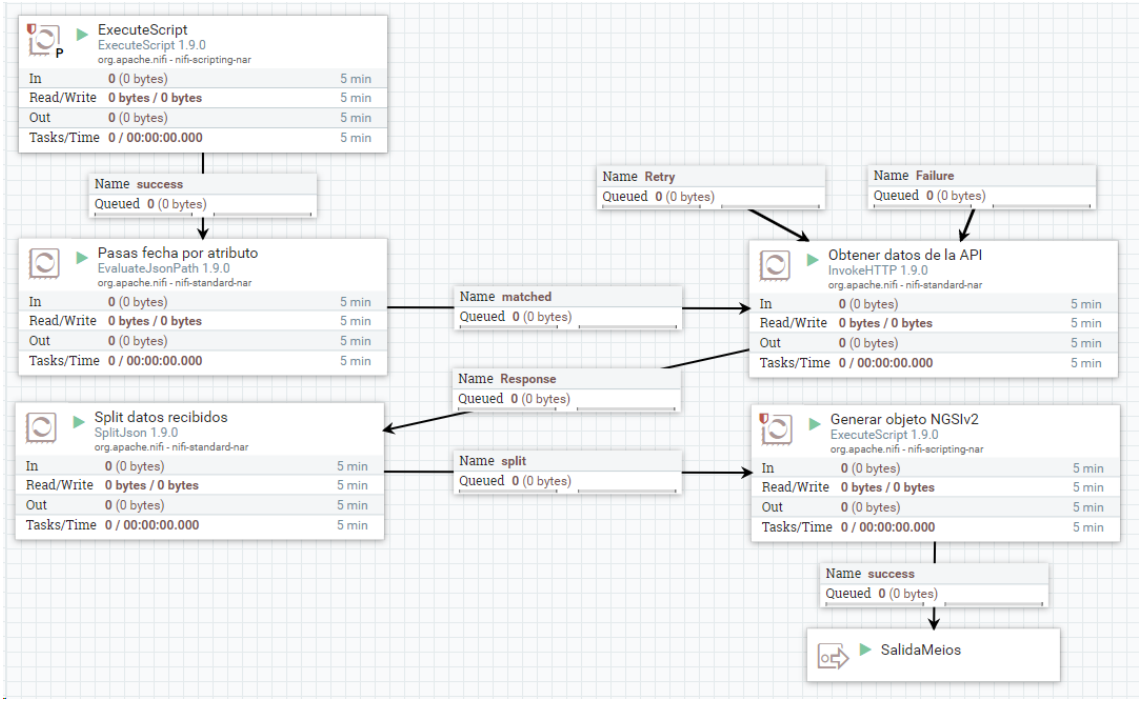

Ilustración 26. Flujo para obtener medios humanos PM

En la imagen anterior se puede ver una parte principal del flujo que se encarga de insertar los medios humanos PM diariamente en nuestra base de datos MongoDB para que sean visualizados en el *dashboard* que ha de desarrollarse.

El proceso[18] que sigue el flujo anterior es el siguiente:

1. **ExecuteScript**: es un proceso que nos permite ejecutar un script en un lenguaje que escojamos de los posibles (Clojure, ECMA, Groovy, Phython y Ruby). En nuestro caso escogemos ECMA ya que vamos a implementar un script en JavaScript. Este script será el encargando de generar la fecha que necesita el proceso "Obtener datos de la API" para pasarla por parámetro en la URL. La fecha que generamos es la del día siguiente al que se ejecuta porque es necesario hacer disponibles esos datos a las 23:45. El formato que debe cumplir la fecha es el siguiente DD/MM/YYYY. Este es el punto de partida del flujo, que como se puede ver en la imagen está planificado con un cron que se ejecuta diariamente a las 23:45.

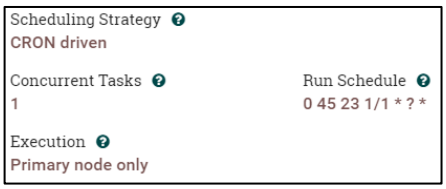

Ilustración 27. Configuración de la planificación del flujo

2. **Pasar fecha por atributo**: el fichero que se pasa de proceso a proceso (llamamos proceso a cada caja del flujo) se llama *flowfile*. Este *flowfile* tiene un contenido y unos atributos. Para que se pueda coger la fecha por parámetro debe estar en los atributos y no en el contenido del *flowfile*, que es como se genera, por lo que es necesario este proceso para añadirlo a los atributos. Esto es así, ya que el contenido del *flowfile* se envía cuando es una petición de tipo POST, al ser un GET no se envía contenido por lo que es necesario pasar esa fecha como parámetro de la URL del servicio al que debemos atacar. En la imagen se ve la configuración que debe tener para coger la fecha.

Property	Value
Destination	flowfile-attribute
Return Type	auto-detect
Path Not Found Behavior	ignore
Null Value Representation	empty string
fecha	$.fecha

Ilustración 28. Configuración para coger atributos del flowfile

3. **Obtener datos de la API**: en este proceso se hace la petición para obtener los datos. El método utilizado es GET, y la URL que se debe atacar es la siguiente: *${smartcity.pm.medios.url}=${fecha}&numMaxTuplos=-1*.
    a. ${smartcity.pm.medios.url}: esta es la URL que debe atacar configurada en un fichero de *properties*. Está externalizado para tener un fichero por entorno ya que se desarrolla con datos de prueba.
    b. ${fecha}: este es el atributo del *flowfile* que hemos creado con la fecha generada, para la que se desea traer los medios.
    c. numMaxTuplos=-1: se le pasa -1 como parámetro para que devuelva todos los medios humanos que haya cargados y no una cantidad exacta ya que perderíamos información, medios humanos PM.
4. **Split datos recibidos**: en este proceso obtenemos el *flowfile* del proceso anterior que contiene la respuesta de la API a la petición que le hemos enviado. Aquí separamos en *flowfiles* independientes todos los medios humanos PM para que el siguiente proceso los prepare para la inserción en la base de datos.
5. **Generar objeto NGSIv2**: éste es el proceso más importante, que es transformar el objeto con la información proveniente del servicio al formato NGSIv2, que es el estándar que requiere Orion para almacenar la información.
6. **SalidaMedios**: este proceso es el que conecta este flujo con otro que es el encargado de insertar las entidades creadas en la base de datos.

En la siguiente imagen, *ilustración 29*, se puede ver el proceso en el que se trata la entidad antes de pasarla al proceso del Orion, y posteriormente la llamada a Orion para realizar la inserción.

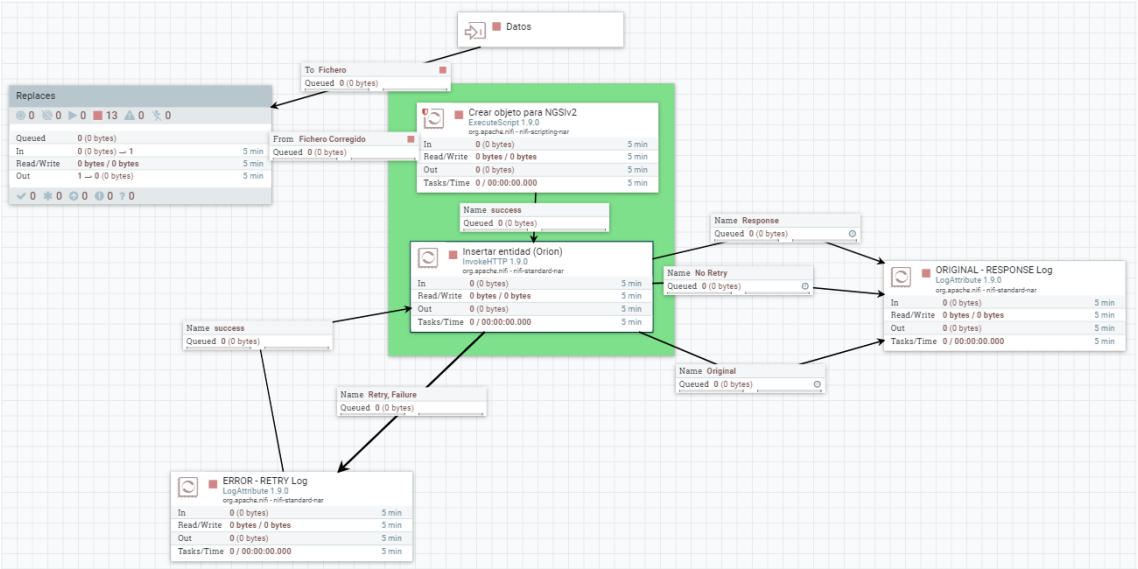

Ilustración 29. Flujo para insertar entidades en Orion

El flujo sigue el siguiente orden:

1. **Datos**: proceso de entrada, conectado el flujo anterior por el que entran los datos de los medios en formato NGSIv2.
2. **Replaces**: grupo de procesos en el que se realiza un reemplazo de caracteres que no son aceptados por Orion. Si no se hiciese este paso muchas de las entidades que se enviasen a insertar fallarían y perderíamos mucha información.
3. **Crear objeto para NGSIv2**: aquí, la información que llega en el flowfile está en formato NGSIv2. Sin embargo, ese objeto hay que insertarlo en otra estructura, en formato JSON también, en el que se le indica la operación que debe realizar. En nuestro caso es un APPEND, cuya función es la de insertar la entidad si no existise y en caso contrario actualizarla con los datos que se le pasen.
4. **Insertar entidad (Orion)**: este proceso hace la llamada a Orion, y como se comentaba anteriormente, no es necesaria una URL expuesta para realizar la petición, sino que se puede hacer por nombre de contenedor. La URL que se configura en el proceso para insertar/actualizar la entidad seria la siguiente: http://orion:1026/v2/op/update.

De esta forma, cuando termina la ejecución de las dos partes del flujo, tendremos insertados los medios humanos pertenecientes al siguiente día para poder trabajar con ellos y consultar los datos necesarios.

B. Vista

Esta parte de la implementación se centra en la iteración del usuario con el módulo a desarrollar. Aquí se ha utilizado una de los *frameworks* más populares en la actualidad, AngularJS.

Angular hace llamadas AJAX, desde el *frontend*, a nuestra API implementada en el servidor Node.js. Éste, hará la petición a la base de datos (BD), si fuese necesario, ya sea para modificar una entidad o para consultarla. La BD devuelve el objeto con la petición que se ha realizado, sobre el cual, el controlador realiza las acciones que se considere necesarias para que la información se visualice en la pantalla de forma correcta.

Antes de continuar es conveniente hablar de las **principales características**[8][9] que tiene este *framework*, entre las que destacan:

- **2-Way-data-binding**: esta es una de las características más importantes, hace referencia a que se están observando continuamente cambios, tanto en la vista como en el controlador. Así, cuando en uno se produce un cambio, el otro es notificado y viceversa.

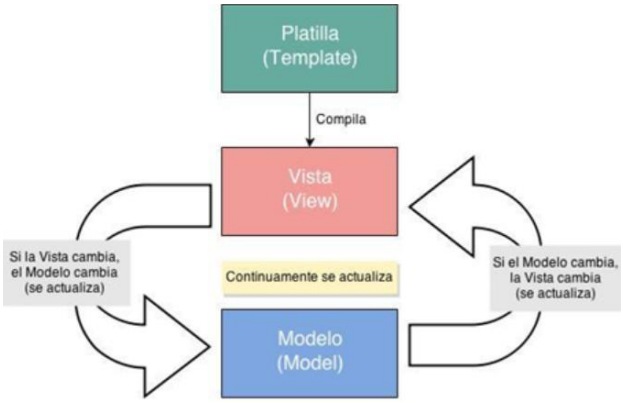

Ilustración 30. Flujo de bindeo AngularJS

- **Expresiones**: los valores que queremos que se muestren en la vista deben ser puestos entre llaves dobles: {{expresiones o valor}}. Algunos ejemplos:
  - {{1+2}}: reflejará 3.
  - {{a+b}}: reflejará el valor de a más el valor de b.
  - {{user.name}}: reflejará el valor que viene de un atributo en un objeto.
  - {{item[index]}}: reflejará el valor que se encuentra dentro de un arreglo.
- **Directivas**: se podrán crear directivas personalizadas. Estas directivas son elementos HTML, y es la manera correcta de manipular

el DOM. Con estas directivas se puede añadir comportamientos específicos a cada nuevo elemento HTML o directiva.

- **Vistas y rutas**: una aplicación web se compone de muchas páginas HTML. En el caso de Angular se dispone de un HTML, como si fuese una SPA, que hace las funciones de contenedor de HTML. Las páginas que se quieran mostrar de la aplicación se mostrarán en esta, de forma que no sería necesario hacer una recarga completa de la página si no que se cambiaría únicamente el contenido del contenedor.

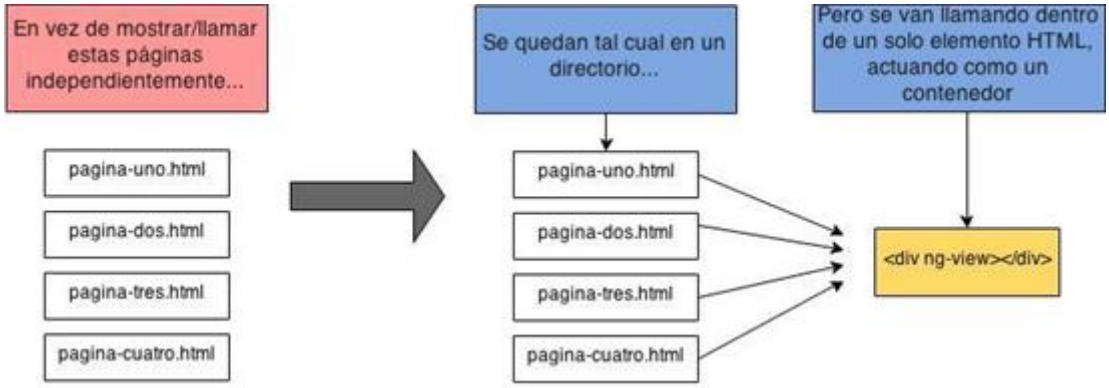

Ilustración 31. Ejemplo de cómo se muestran las vistas

Además de utilizar este *framework*, se utilizan otras librerías que permiten añadir dinamismo y usabilidad, como son:

1. Jquery-clockpicker
2. Bootstrap
3. Boottrap-multiselect

C. Controlador

El controlador se encarga de realizar toda la lógica de negocio del módulo desarrollado. El patrón seleccionado permite individualizar los controladores que contendrán la lógica de las dos vistas del módulo. De forma que, si esta funcionalidad se quiere llevar a otra plataforma que tenga la misma arquitectura, se podría implantar sin ningún problema.

A continuación, se detallan las funciones que cumple cada controlador y una breve explicación de cada una, para poder asociar los requisitos funcionales a las funciones desarrolladas.

ControlMediosPMController

Lo primero que se hace aquí es una consulta a la base de datos, a través del Fiware de Orion, para obtener los medios humanos del día y mostrarlos en formato lista en el *dashboard*.

```
GetMediosHumanos: function (date) {
  var deferred = $q.defer();

  var init = moment().startOf("day");
  var startTimestamp = init.unix() * 1000;
  var endTimestamp = init.add(1, "days").unix() * 1000;

  if (date) {
    startTimestamp = date.unix() * 1000;
    endTimestamp = date.add(1, "days").unix() * 1000;
  }

  var filter = [];

  filter.push({
    "type": "FIWARE::StringQuery",
    "value": "finTurno>='" + startTimestamp + "';inicioTurno<'" + endTimestamp + "'"
  });

  Api.GetEntitiesFiltered("PM.medios", filter, 1, 1000).then(function (response) {
    var data = parseMedios(response.data, startTimestamp);
    deferred.resolve(data);
  }).catch(function (error) {
    deferred.reject(error);
  });

  return deferred.promise;
},
```

Ilustración 32. Función de llamada a la API

En la imagen se puede ver la petición que se hace a la API, en la que se filtra por la fecha de los medios que queremos mostrar. Está configurado como una promesa para no perder la sincronía del flujo de carga, ya que se trata de una petición asíncrona y proveen de *callbaks*, en vez de pasárselo, para que se realicen las acciones que sean necesarias con la respuesta de la petición. En el caso de la *ilustración 32*, se puede ver que se espera a que la petición de consulta a la base de datos (BD) responda con el resultado, y con este se realizan una serie de acciones sobre los datos obtenidos y se devuelven a otro *callback* que estará esperando esa información transformada.

Además, este controlador tiene funciones para aplicar los filtros, especificados en los requisitos funcionales, y otras funciones especificadas a continuación:

a. **Tipo de medio**: se puede aplicar un filtro para que aparezcan listados únicamente los medios de un tipo concreto, entre los que se encuentran: oficiales de policía, inspectores, control de tráfico, grúas y motociclistas.

b. **Inicio de turno**: a través de un *clockpicker,* que es un elemento de Bootstrap para introducir horas en un input, se puede elegir una hora por la que se filtrarán por inicio de turno los medios cargados.
c. **Medios activos en cierta hora**: también, a través de un *clockpicker,* que es un elemento de Bootstrap para introducir horas en un input, se puede escoger una hora, y se mostrarán los medios humanos que en esa hora estén activos.
d. **Búsqueda por palabra clave**: a través de un input de texto se pueden filtrar medios por palabras que contengan alguno de sus atributos. Esto puede servir, por ejemplo, para buscar medios humanos de un indicativo de radio concreto, o buscar un medio concreto por su número de placa.
e. **Mostrar medios de una fecha determinada**: a través de un *datepicker,* que es un elemento de Bootstrap para introducir fechas en un input, se puede elegir una fecha anterior, o posterior si ya han sido cargados los medios futuros, para que se listen en la tabla esos medios concretos.
f. **Añadir nuevo medio**: un botón que nos abra un modal y nos permita rellenar un formulario para añadir un medio que no ha sido cargado, a través del flujo de Apache NiFi. Se permite cargar el medio tanto en la fecha actual como en fechas anteriores y posteriores. Se han de rellenar todos los campos obligatorios para que el medio pueda ser cargado.
g. **Limpiar filtros**: botón que permita eliminar todos los filtros que hayan podido ser aplicados y mostrar en la tabla todos los medios humanos de esa escala.
h. **Editar medio en modal**: haciendo clic en el servicio diario asignado de un medio de la lista, se abre un modal con los datos de ese medio que se pueden modificar.
i. **Editar medio en tabla**: se requiere que algunos de los campos que se muestran en la tabla se puedan modificar en ella. Como puede ser: activar el medio, cambiar el inicio y fin de turno, modificar el indicativo, el número de radio y el nombre del medio.
j. **Guardar cambios**: todos los cambios que se hayan producido sobre tabla directamente requieren de esta acción. Al contrario que si se modifica el medio en el modal que para que los cambios se persistan es necesario guardar desde el modal.
k. **Imprimir**: botón que permite imprimir el listado completo de lo que se está visualizando en ese momento.

Ilustración 33. Filtros y acciones aplicables a los medios listados

GestionMediosPMController

Para que aquí podamos realizar las funciones especificadas en los requisitos funcionales, es necesario activar los medios previamente en la vista de medios humanos activos.

Una vez activados, se mostrarán los medios humanos en sus respectivas tablas, en función del tipo de medio humano que sea.

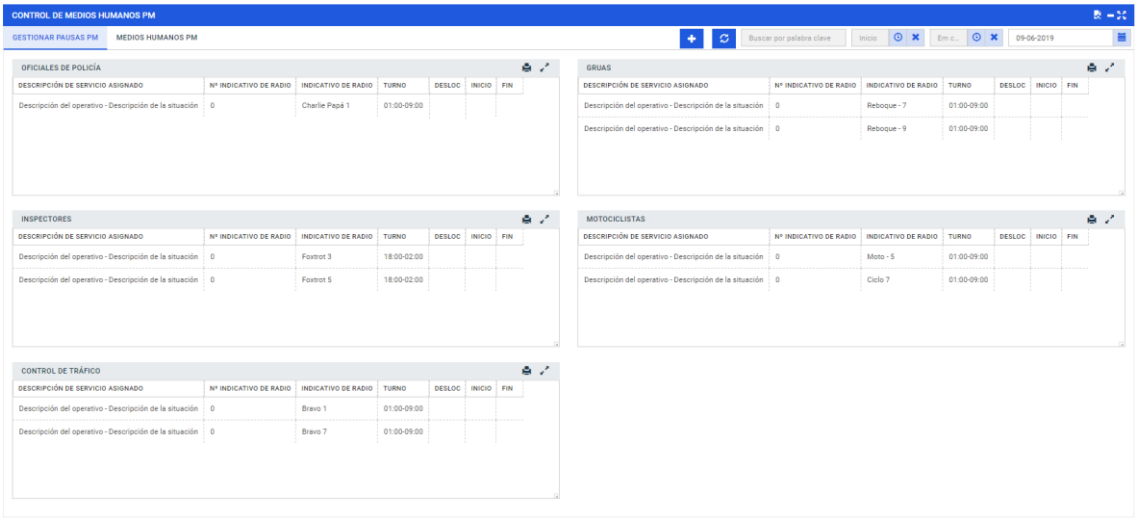

Ilustración 34. Vista gestión medios humanos PM

En la imagen anterior se ven los medios humanos activos ubicados en sus respectivas tablas. Una vez listados se pueden empezar a manejar las pausas con la lógica especificada en el requisito funcional **RF007**. Sobre estos datos también se pueden aplicar algunos de los filtros comentados en el punto anterior. Estos filtros son: inicio de turno, medios activos en cierta hora, mostrar medios de una fecha determinada, búsqueda por palabra clave y limpiar filtros.

Lo más destacable y el núcleo de este proyecto es esta vista, la que lleva la lógica más compleja. Esto es debido a la necesidad de que los cambios producidos por un funcionario en su ordenador tienen que ser visualizados en la vista de cualquier otro funcionario que la tenga abierta en tiempo real.

Esto, se ha conseguido gracias a los **WebSockets**[13], que permite mantener una conexión continua entre el cliente (Navegador web) y el servidor. En la parte del cliente se ha implementado una función que está continuamente escuchando. Cuando recibe datos a través del canal configurado, se lanza una función, *onmessage*, que recibe la información que se ha enviado por el canal. Esta función, recibe y trata los datos y los manda a través de un *broadcast* para que lo reciban todos los usuarios que estén conectados a la plataforma. Sin embargo, solo aquellos que estén visualizando el *dashboard* de control de medios humanos PM y estén

en la vista de gestión de pausas, recogerán esa información y se mostrará en sus pantallas en tiempo real.

Lo anteriormente comentado se refiere a la parte del cliente. En la parte del servidor, se manda la información a través del *websocket* gracias al *framework* **ExpressJS**. Gracias a este *framework*, podemos obtener los usuarios que están conectados a la plataforma, y enviar a través del canal configurado para el *websocket* la información individualizada.

```
// configuramos el canal
var aWss = expressWs.getWss(config.app.base_url + '/websocket');
// recorremos los usuarios conectados a la plataforma
aWss.clients.forEach(function (client) {
  // lógica propia de la gestión de pausas
  pausa[property] = (value == "3" ? "" : value);
  // tipo de mensaje que recibe el cliente a través del websocket
  pausa.typeMessage = "pausasOverload";
  // indica la propiedad del cambio, si es para deslocalización,
inicio o fin.
  pausa.property = property;
  // objeto del medio humano modificado
  pausa.data = result.data != "" ? result.data : "";
  // comprobamos que la informacion a pasar es un objeto y lo
transformamos a string para pasarlo por el socket
  if (typeof pausa == "object") {
    pausa = JSON.stringify(pausa);
  }
  // se envía la información al cliente
  client.send(pausa);

});
```

Ilustración 35. Código ejemplo envío datos por socket backend

Además de la lógica implementada para cubrir el requisito funcional RF007 con las tecnologías anteriormente mencionadas, hay que explicar otras funciones que deben cumplirse en esta vista, además de los filtros ya explicados, para cada tabla en función del tipo de medio humano:

- **Imprimir**: imprime la información representada en la tabla seleccionada.
- **Fullscreen**: visualiza la tabla en pantalla completa.
- **Deslocalización**: inicia la deslocalización
- **Inicio**: inicia la pausa
- **Fin**: finaliza la pausa

Por último, es necesario hablar de cómo se notifica al usuario que gestiona las pausas de los medios humanos PM que se ha excedido el tiempo límite de deslocalización y de pausa. En el momento en el que un medio humano PM notifica la deslocalización o el inicio de la pausa, se planifica una tarea a futuro, que se ejecuta si no ha cambiado de estado la pausa en un tiempo preconfigurado. Este tiempo es de 20 minutos en ambos casos, como se especifica en el requisito

funcional. Sin embargo, es configurable a través de *backoffice* por un usuario que tenga los permisos para hacerlo, este desarrollo no entra dentro del ámbito de desarrollo de este proyecto.

D. Modelo

Esta capa es la que especifica cómo se realiza el acceso a la información, a la base de datos, para el módulo que nos atañe.

En este caso, se hace uso de un **Fiware** que recibe el nombre de **Orion Context Broker**. Como se ha comentado en el apartado de tecnologías, *Orion Context Broker es un importante componente que permite administrar el ciclo de vida de los datos en el conjunto Fiware creado por Telefónica. Así, a través de este context broker es posible tanto añadir, actualizar como consultar entidades.*

Este Fiware ofrece una capa de abstracción entre la aplicación y la base de datos. Utiliza un modelo de datos denominado NGSIv2, que define una entidad con formato JSON que se forma con los siguientes atributos obligatorios: id y type. Al almacenar estas entidades en una base de datos no relacional, MongoDB, el atributo type haría las veces de nombre de la tabla en una base de datos relacional, y el atributo id de primary key, PK. Además de estos dos atributos, se pueden añadir todos los atributos que se consideren necesarios para los diferentes tipos de entidad.

En la *ilustración 37* se puede ver un ejemplo de petición al Fiware de Orion. De este ejemplo hay que destacar algunos puntos:

- En la uri se puede ver que el host al que se ataca es inusual. Esto es debido a que al trabajar con contenedores de Docker se pueden hacer referencia a ellos directamente, sin la necesidad de exponer ese contenedor.
- La petición es de tipo POST ya que necesitamos enviar la entidad que hay que insertar o actualizar.

```
{
      "actionType": "APPEND",
      "entities": [{
                  "id": "PK de la entidad",
                  "type": "tipo de la entidad, 'nombre de tabla'",
                  "atributoX": {
                        "type": "String",
                        "value": "valor del atributo"
                  }
            }]
}
```

Ilustración 36. Ejemplo objeto insertar/actualizar entidad

El servicio update que expone Orion, como se puede ver en la *ilustración 37*, requiere que se le pase un objeto como el de la *ilustración 36* en el que se le especifica el tipo de acción que el Fiware debe llevar a cabo. La más utilizada es la que se ve en la *ilustración 36*, APPEND. Esta acción indica que la entidad/es que estén en el atributo entities, se inserten en la base de datos. En caso de que esas entidades ya estuviesen insertadas, se actualizarían los atributos con los valores que se especifiquen. Hay que destacar que no es necesario enviar la entidad completa. Por ejemplo, si queremos actualizar un único atributo de la entidad, habría que componer un objeto con solo ese atributo y, por supuesto, el id y el type que son valores obligatorios en todos estos tipos de peticiones.

```
function getValue(data, path, isKeyValue) {
    var uri = "http://orion:1026/v2/op/query";
    if (isKeyValue) {
        uri = "http://orion:1026/v2/op/query?options=keyValues";
    }
    var options = {
        method: 'POST',
        headers: {
            'User-Agent': 'Request-Promise',
            'Content-Type': 'application/json',
            'Accept': 'application/json',
            'fiware-service': 'fs_data',
            'fiware-servicepath': path
        },
        rejectUnauthorized: false,
        uri: uri,
        body: data, // JSON con la información y la acción a
ejecutar
        json: true // Automatically stringifies the body to JSON
    };

    var deferred = q.defer();

    rp(options).then(
            function (response) {
                deferred.resolve(response);
            },
            function (error) {
                deferred.reject(error);
            }
        );
    return deferred.promise;
}
```

Ilustración 37. Ejemplo de llamada al Fiware

5. PRUEBAS

En este capítulo se van a explicar las pruebas realizadas en el módulo para comprobar su correcto funcionamiento y asegurarnos de cumplir los requisitos. Es necesario destacar, que aparte de las pruebas recogidas en los siguientes puntos, existe un equipo de trabajo compuesto por dos personas que verifican que todos los desarrollos funcionan de forma correcta, este equipo se denomina QA.

5.1. Pruebas unitarias

Las pruebas unitarias consisten en verificar que cada capa de nuestro patrón MVC cumplan con los requisitos definidos y el comportamiento sea el correcto. En este caso se va a comprobar que la lógica de aplicación de nuestro componente funcione correctamente.

Para esto, se va a utilizar un *framework* que nos ofrece un marco de pruebas JavaScript para aplicaciones desarrolladas con Node.js llamada MochaJS.

Los resultados obtenidos de las pruebas desarrolladas son los siguientes:

```
restify listening at XXXXXXXXXX
  mediosHumanosPM
    GET
      ✓ Devuelve los medios humanos PM para el dia actual.
      ✓ Devuelve los medios humanos PM para el día anterior.
    POST
      ✓ Guarda los cambios de un medio humano PM.
      ✓ Añade un nuevo medio humano PM.
      ✓ Modifica los valores de pausa de un medio humano PM.


  5 passing (4365ms)
```

Ilustración 38. Resultados pruebas unitarios, MochaJS

5.2. Pruebas de integración

Estas pruebas confirman el correcto funcionamiento de los módulos de los que está formado el componente.

Se han ido generando mientras se iban desarrollando las diferentes capas del componente. Lo que se ha probado es la interacción entre las diferentes capas:

- Capa de persistencia con Orion, base de datos.
- Capa de negocio con la capa de persistencia.
- Controladores con la capa de persistencia.

Verificada la interacción entre esas capas, hay que probarlo integrado en la plataforma, con la vista montada. Para ello, se ha añadido todo el código de nuestro componente en la plataforma en la que lo debemos implantar, y se han ido probando todas las funcionalidades desarrolladas.

5.3. Pruebas de sistema

Completadas las pruebas anteriores se procede a realizar las pruebas de sistema. Estas se basan en comprobar el sistema completo y verificar que se cumplen los requisitos nos funcionales. En este caso se harán únicamente pruebas de seguridad ya que se trata de un componente a integrar en un sistema que ya debe cumplir los requisitos no funcionales definidos para la plataforma.

A. Pruebas de seguridad

Las pruebas de seguridad, en este caso, tiene como objetivo que la visualización y la información del componente sea accesible únicamente por usuarios que tengan el rol correcto para poder acceder a ello.

Configurado todo, y creado un usuario con el perfil correcto, se comprueba que tiene acceso al *dashboard* nuevo y a la información que este hace disponible. También se ha accedido a la plataforma con otro usuario que no tenga ese perfil y, como era de esperar, no tiene acceso al *dashboard* ni a la información.

5.4. Pruebas de aceptación

Las pruebas de aceptación son la etapa final del proceso de pruebas, paso intermedio entre el desarrollo del módulo y la puesta en marcha de éste para su uso operacional. Miden la satisfacción del cliente con el sistema y comprueban que se han cumplido los objetivos especificados en los requisitos funcionales.

Durante el desarrollo del módulo se han ido enseñando partes al cliente que han ido verificando cada desarrollo. De forma que cuando se han concluido el desarrollo ya se habían satisfecho muchos de los requisitos funcionales y se aprobó, por parte del cliente, la puesta en producción del módulo para comenzar su uso por los usuarios adecuados.

6. CONCLUSIONES Y TRABAJOS FUTUROS

Este último capítulo congrega las conclusiones recogidas del desarrollo de este proyecto y las características que pueden ser incluidas en un futuro si el cliente lo requiriese.

6.1. Conclusiones

Este trabajo de fin de grado, TFG, recoge las diferentes etapas de las que se compone el desarrollo de un nuevo módulo para una plataforma ya existente. Se hace hincapié en el análisis, diseño e implementación de este módulo para satisfacer los requisitos funcionales impuestos por el cliente. Para llevar a cabo el módulo se ha seguido una metodología de desarrollo iterativa incremental, generando iteraciones por cada una de las 3 partes de las que se compone el módulo y se ajustan a las etapas típicas del ciclo de vida de desarrollo del software, que son: análisis de requisitos, diseño, implementación, pruebas y documentación.

La realización del proyecto descrita anteriormente ha permitido cumplir satisfactoriamente los objetivos planteados, que eran desarrollar un módulo de gestión de pausas de medios humanos PM que sustituyese la forma de trabajo anterior y llevar una mejor metodología.

Personalmente, he aprendido tecnologías nuevas, tanto con el uso de Apache NiFi con los WebSockets en JavaScript. Me ha ayudado a mejorar profesional e internamente en la empresa, siguiendo la misma disciplina que llevaba aplicando durante los años anteriores, pero con más ganas de hacer mejor las cosas si cabe.

6.2. Trabajos futuros

El contrato con el cliente se ha estructurado en dos fases, esta primera en la que se crea este módulo para comenzar a digitalizar estas acciones que se llevaban a cabo manualmente; y otra en la que se definen una serie de objetivos a cumplir. En esta segunda fase, se han de añadir las siguientes funcionalidades:

- **Activar medios automáticamente**: planificar una tarea que se ejecute al inicio de los turnos y active y desactive automáticamente los turnos sin tener que hacerlo de forma manual. Esto permitiría disminuir el tiempo que un usuario requiere para manejar las pausas de los medios humanos PM que estén en su turno en ese momento.
- **Añadir filtro para visualizar únicamente medios humanos PM activos**: a parte de los filtros ya configurados, habría que añadir un

filtro que permita visualizar en la tabla únicamente los medios que estén activos.

- **Edición en simultáneo de los medios humanos PM**: éste es el caso más complejo de los requerimientos del cliente. Como varios usuarios pueden modificar datos a la vez, estos cambios tienen que estar visibles para el resto de los usuarios que tengan abierto este *dashboard*. Por lo que los cambios que haga uno, deben estar disponibles en el otro en tiempo real para que no se produzcan incongruencias en los datos mostrados en cada momento. Con esto se consigue que dos usuarios conectados no estén visualizando contenidos diferentes.

BIBLIOGRAFÍA

[1] Sommerville, I. 2011. *Ingeniería del Software*, 9a Edición. Addison-Wesley. ISBN 978-607-32-0603-7.

[2] Instituto de ingeniería del conocimiento. 2016. [Consulta: 20 mayo 2019]. Disponible en: http://www.iic.uam.es/innovacion/big-data-caracteristicas-mas-importantes-7-v/

[3] LOGICALIS: a blog from business and technology working as one. 2018. [Consulta: 20 mayo 2019]. Disponible en: https://blog.es.logicalis.com/analytics/las-tres-v-de-big-data-analytics-m%C3%A1s-vigentes-que-nunca

[4] ORACLE. 2019. *¿Qué es Big Data?* [Consulta: 20 mayo 2019]. Disponible en: https://www.oracle.com/es/big-data/guide/what-is-big-data.html

[5] FIWARE-ORION. 2019. *Orion Context Broker.* [Consulta: 26 mayo 2019]. Disponible en: https://fiware-orion.readthedocs.io/en/master/index.html

[6] TELEFONICA. 2015. FIWARE, Internet de las cosas y conectividad. [Consulta: 26 mayo 2019]. Disponible en: http://www.tidchile.cl/tecnologia

[7] TELEFONICA. 2018. FIWARE-NGSIv2. [Consulta: 26 mayo 2019]. Disponible en: http://telefonicaid.github.io/fiware-orion/api/v2/stable/

[8] Carlos Azaustre: a blog from Tutorial de AngularJS. 2014. [Consulta: 14 junio 2019]. Disponible en: https://carlosazaustre.es/tutorial-ejemplo-de-aplicacion-web-con-angular-js-y-api-rest-con-node/

[9] HTML5FACIL. 2015. Las principales características de Angular. [Consulta: 10 junio 2019]. Disponible en: http://html5facil.com/tutoriales/las-principales-caracteristicas-de-angularjs/

[10] MOZILLA. 2019. HTML5. [Consulta: 28 mayo 2019]. Disponible en: https://developer.mozilla.org/es/docs/HTML/HTML5

[11] MOZILLA. 2019. JavaScript. [Consulta: 28 mayo 2019]. Disponible en: https://developer.mozilla.org/es/docs/Web/JavaScript

[12] MOZILLA. 2019. CSS3. [Consulta: 28 mayo 2019]. Disponible en: https://developer.mozilla.org/es/docs/Web/CSS/CSS3

[13] MOZILLA. 2019. Escribiendo aplicaciones con WebSockets. [Consulta: 14 junio 2019]. Disponible en: https://developer.mozilla.org/es/docs/Web/API/WebSockets_API/Writing_WebSocket_client_applications

[14] PUTTY. 2019. *PUTTY* [programa de ordenador]. [Usado: 30 mayo 2019]. Disponible en: https://www.putty.org/

[15] NODEJS. 2019. *NODEJS* [programa de ordenador]. [Usado: 30 mayo 2019]. Disponible en: https://nodejs.org/

[16] AngularJS. 2018. AngularJS. [Consulta: 28 mayo 2019]. Disponible en: https://angularjs.org/

[17] WinSCP. 2019. *WinSCP* [programa de ordenador]. [Usado: 30 mayo 2019]. Disponible en: https://winscp.net/eng/index.php

[18] HORTONWORKS. 2017. ExecuteScript Cookbook. [Consulta: 10 junio 2019]. Disponible en: https://community.hortonworks.com/articles/75032/executescript-cookbook-part-1.html

[19] Pavón Mestras, Juan. 2009. *El patrón Modelo-Vista-Controlador.* Dep. Ingeniería del Software e Inteligencia Artificial. Madrid: Universidad Complutense de Madrid (UCM). [Consulta: 13 junio 2019]. Disponible en: https://www.fdi.ucm.es/profesor/jpavon/poo/2.14.MVC.pdf

[20] Centro Universitario de Desarrollo Intelectual. 2017. Orion Context Broker. [Consulta: 03 junio 2019]. Disponible en: http://www.cudi.edu.mx/boletin/2017/FIWARE_Taller/README_JCarlosUrteaga.pdf